Mikrokomputery

John Lansdown

GRAFIKA KOMPUTEROWA

Grafika komputerowa

John Lansdown to znany międzynarodowy ekspert, który specjalizuje się w grafice komputerowej od 1960 roku. Jest twórcą graficznych efektów do trzech długometrażowych filmów i przeszło stu telewizyjnych: reklamowych i dokumentacyjnych. Był wykładowcą w Royal College of Art i St Martins School of Art w Londynie. Obecnie pełni funkcję prezesa System Simulation Ltd w Londynie.

Mikrokomputery

John Lansdown

Grafika komputerowa

Z języka angielskiego przełożyli

Jan Zabrodzki, Marian Łakomy

Wydawnictwa Naukowo-Techniczne
Warszawa 1990

Dane o oryginale
Computer Graphics (Teach Yourself Books)
John Lansdown
First published 1987 in the English language by
Hodder and Stoughton Ltd, Sevenoaks, Kent, England

Redaktor *Jadwiga Strzelecka*
Przygotowały do druku *Marta Jeczeń, Jadwiga Strzelecka, Mariola Tyburska*
Projektant serii *Janusz Rybicki*
Okładkę i strony tytułowe projektował *Andrzej Pilich*

681.32

Książka zawiera zbiór podstawowych wiadomości z grafiki komputerowej. Przedstawiono urządzenia wejściowe i wyjściowe, omówiono cztery podstawowe działania w grafice, tj. modelowanie, transformację, obcinanie i wyświetlanie w przestrzeni dwuwymiarowej i trójwymiarowej. Książka zawiera wiele przykładów podstawowych programów graficznych. Sposób przedstawienia zagadnień i przypomnienie elementarnych podstaw matematyki sprawiają, że książka jest przeznaczona dla szerokiego kręgu czytelników.

ISBN 83-204-1229-3

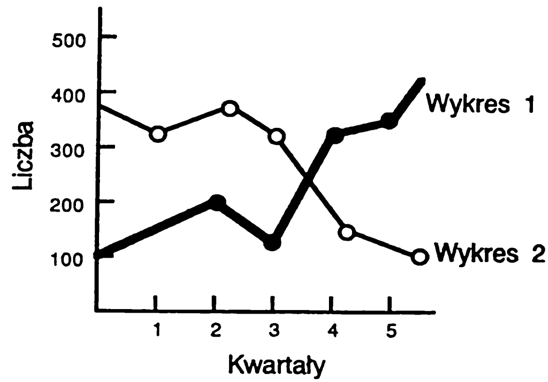
Spis treści

1. Wstęp

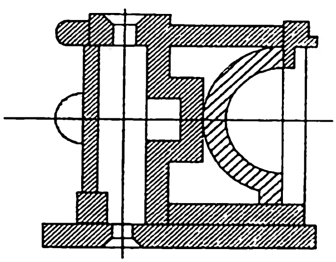
Znaczenie grafiki komputerowej

Prawdopodobnie najbardziej spektakularnym i nieoczekiwanym wynikiem rozwoju informatyki stało się w ostatnich latach użycie komputerów do tworzenia graficznych obrazów różnego rodzaju. Wiele filmów reklamowych wykorzystuje grafikę komputerową i sekwencje animowane za pomocą komputera. Prawie codziennie na ekranie telewizorów pojawiają się nowe obrazy wytworzone za pomocą komputera. Często są to obrazy o tak wysokiej jakości i realiźmie, że nie podejrzewamy nawet udziału komputera w ich tworzeniu. Powstała cała nowa dziedzina przemysłu zaspokajająca zaistniałe potrzeby, zatrudniająca projektantów, programistów, pracowników badawczych i producentów. Do uzyskania graficznych obrazów wykorzystuje się najpotężniejsze komputery świata.

Obrazów graficznych nie tworzy się dla samego ich tworzenia. W gospodarce, grafika komputerowa jest używana do pokazywania tendencji rozwojowych, reprezentacji danych statystycznych (rys. 1.1) oraz do wspomagania kierownictwa przedsię-



Rys. 1.1. Rysunek w statystyce



Rys. 1.2. Rysunek z programu typu CAD

biorstwa w podejmowaniu decyzji. W architekturze i budownictwie rysunki zawsze odgrywały kluczową rolę w przekazywaniu realizatorom pomysłów projektowych. Nic więc dziwnego, że projektowanie wspomagane komputerem (CAD), mające znakomite możliwości graficzne (rys. 1.2), jest w powszechnym użyciu. W nauce i medycynie grafika komputerowa jest coraz częściej używana przy wyjaśnianiu skomplikowanych zjawisk. Nawet w domowym zastosowaniu komputerów grafika - często o znakomitej jakości - odgrywa dużą rolę w grach i w programach do nauczania. Tak więc, nie ma takiej dziedziny ludzkiej działalności, w której nie występuje grafika komputerowa.

Rysunki, diagramy i obrazy ułatwiają zrozumienie wszelkiego rodzaju informacji. Pomysły można przedstawić jaśniej i bardziej przekonywująco w formie graficznej, ale przecież nie każdy ma niezbędne do tego umiejętności. Pomaga w tym grafika komputerowa, umożliwiająca skoncentrowanie się na przekazywaniu informacji.

Cel książki

Celem książki jest zapoznanie użytkowników komputerów z metodami i narzędziami programowymi grafiki komputerowej. Po przeczytaniu tekstu, napisaniu dołączonych programów i wykonaniu ćwiczeń Czytelnik opanuje podstawy dziedziny grafiki komputerowej. Zależnie od dostępnego sprzętu będzie mógł stosować przedstawione w książce techniki wykonywania rysunków o pożądanej złożoności i użyteczności. Dziedzina grafiki komputerowej rozwija się szybko. Aby nie pozostać w tyle za ostatnimi osiągnięciami i stale zwiększać swoje umiejętności programowania jest niezbędne czytanie książek i artykułów. Dokładne przeczytanie tej książki ułatwi Czytelnikowi zrozumienie trudniejszych publikacji.

Wydruki podprogramów

Przeglądając programy w rozdziale 7 i dalszych można zauważyć, że nie są one napisane w żadnym ze znanych języków programownia. Są one napisane w tak zwanym *pseudokodzie*, który jest zbliżony do języka programowania, ale w podanej postaci nie może być bezpośrednio wykonany w żadnym komputerze. Używamy go do pisania programów, które przy niewielkim wysiłku można zrozumieć i przekształcić na wybrany język programowania. Czasami, w celu lepszego zrozumienia działania programu, nie uwzględniono konstrukcji zwiększających jego szybkość i efektywność; przy kodowaniu należy wziąć to pod uwagę. W Dodatku 2 pokazano jak można uzyskać większą efektywność w różnych językach programowania. Przed przystąpieniem do programowania jest zalecane przejrzenie tego dodatku.

Także w celu lepszego zrozumienia przedstawionych metod, dopuszczono się pewnej dowolności jeśli chodzi o pojęcie równości. Ponieważ komputery nie przechowują liczb rzeczywistych w postaci dokładnej, to próba testowania czy dwie liczby rzeczywiste są takie same, może prowadzić do błędnych wyników. Należy uwzględnić to przy kodowaniu programów i zmieniać wyrażenia typu

If A = B **Then** ...

na postać

If Abs(A–B) < E **Then** ...

w której E jest pewną niewielką liczbą rzeczywistą, na przykład 0.0001.

Nasz pseudokod przypomina Basic strukturalny, Pascal, Comal lub Algol. Nawet ludzie z niewielkim doświadczeniem w programowaniu powinni łatwo zrozumieć podane programy. Jeśli Czytelnik będzie kodował programy w języku strukturalnym posiadającym: konstrukcje typu If-Then-Else, puste parametry w wywołaniu procedur i długie nazwy zmiennych, to nie powinien mieć żadnych kłopotów. Jednakże Basic standardowy nie spełnia tych wymagań i przy jego użyciu należy zachować szczególną ostrożność.

Przy użyciu Basicu, największy kłopot sprawia fakt, że wszystkie zmienne są globalne i jeśli podprogram nada im jakieś wartości, to zostają one zachowane w całym programie. Utrudnia to wprowadzenie uprzednio przygotowanych podprogramów, gdyż mogą one zawierać nazwy zmiennych takie same jak już raz użyte. Dwa możliwe sposoby omijania tego problemu podano w Dodatku 2.

Układ książki

Tekst książki dzieli się logicznie na dwie części: w rozdziałach 1 - 5 zostały omówione ogólne zasady i sprzęt, w pozostałych rozdziałach zostało omówione programowanie graficzne. Pierwsza część zawiera niewiele zadań, natomiast w drugiej jest ich bardzo dużo i Czytelnik powinien sumiennie je rozwiązać. Przy samodzielnej pracy jedyna metoda sprawdzania osiągniętych wyników polega na wykonywaniu testów. Na szczęście, w przypadku grafiki komputerowej, można bezpośrednio zobaczyć efekty swojego działania, ale aby dojść do punktu gdy jest już możliwe wykonywanie rysunków, trzeba na początku włożyć wiele pracy. Zadania i przykłady mają to ułatwić. Nie zniechęcaj się. Twój trud zostanie wynagrodzony.

2. Systemy graficzne

Systemy graficzne, podobnie jak systemy komputerowe ogólnego przeznaczenia, składają się z trzech podstawowych części:
- sprzętu;
- oprogramowania - zbioru instrukcji sterujących sprzętem;
- dokumentacji - opisu sprzętu i oprogramowania, dzięki któremu można w pełni wykorzystać możliwości systemu.

Wprawdzie jest możliwe stworzenie prostych rysunków za pomocą dowolnego systemu komputerowego, to jednak w systemie graficznym wszystkie trzy podstawowe części są ze sobą powiązane tak, że pozwalają na łatwe tworzenie różnych obrazów.

2.1. Podstawowy sprzęt

Sprzętową część systemu graficznego (rys 2.1) stanowią:

Jednostka obliczeniowa określana mianem komputera, a w przypadku komputerów osobistych - i coraz częściej komputerów profesjonalnych - po prostu mikroprocesor.

Pamięć, przechowująca dane i program aż do momentu gdy są potrzebne; często określa się ją jako pamięć pomocniczą lub zewnętrzną.

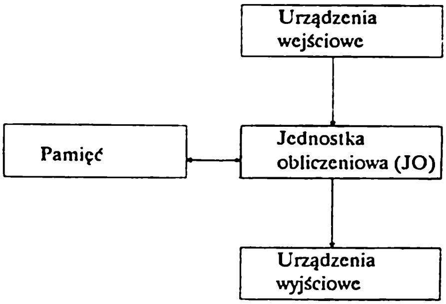
Urządzenia wejściowe przekazywujące informacje graficzne do komputera; np. klawiatura, znajdująca się w każdym systemie; są dostępne także inne, różnorodne urządzenia.

Urządzenia wyjściowe potrzebne do wizualnej prezentacji informacji; np. różnego rodzaju monitory wyświetlające wszelkie postacie rysunków.

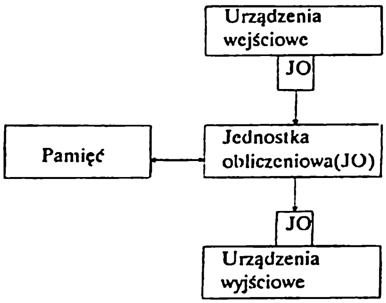
W przypadku niektórych komputerów osobistych i w wielu systemach biurowych urządzenia takie jak: procesor, pamięć (dyskowa lub taśmowa), klawiatura i monitor są dostarczane w postaci zintegrowanej w jedną jednostkę. W innych rozwiązaniach są to jednostki oddzielne, połączone kablami - co pozwala na zakup zestawów dopasowanych do indywidualnych potrzeb użytkownika. W prawie wszystkich

przypadkach, gdy trzeba prowadzić poważne prace graficzne, jest niezbędne posiadanie wielu rodzajów urządzeń wejściowych i wyjściowych. Należy tu ostrzec Czytelnika, że nie jest prostą sprawą dołączenie do komputera urządzeń różnych producentów.

Czasami urządzenia wejścia-wyjścia zawierają procesory umożliwiające wykonywanie funkcji bardziej efektywnie i szybciej. Zmniejszają one obciążenie procesora centralnego; są to *urządzenia inteligentne*. Na rysunku 2.2 przedstawiono konfigurację systemu graficznego zawierającego takie urządzenia.

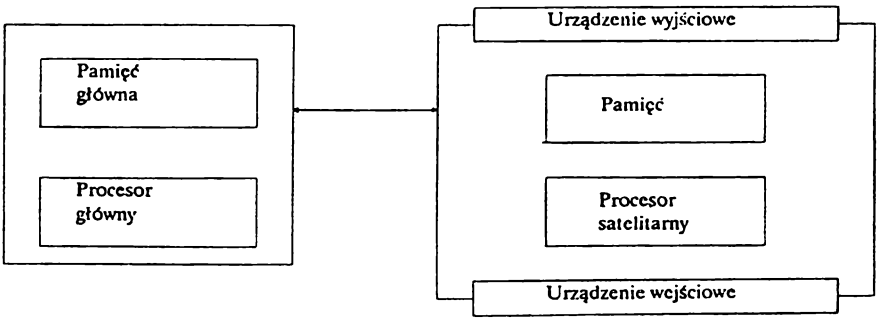


Rys. 2.1. Podstawowa konfiguracja systemu graficznego



Rys. 2.2. Konfiguracja z terminalami inteligentnymi

W niektórych dużych, profesjonalnych urządzeniach graficznych moc obliczeniowa i pamięć są rozdzielone między część główną i satelitarną (rys. 2.3). Część satelitarna zajmuje się wyłącznie obliczeniami graficznymi i współpracą z układem wyświetlania, a część główna zajmuje się innymi obliczeniami i obsługą baz danych. W takiej konfiguracji systemu, komputer główny jest w stanie obsługiwać więcej niż jedną stację satelitarną, która może mieć dołączonych wiele urządzeń wejścia-wyjścia.



Rys. 2.3. Konfiguracja z komputerem satelitarnym

2.2. Podstawowe oprogramowanie

Oprócz sprzętu zasadniczą rolę odgrywają dwa elementy oprogramowania:
- system operacyjny,
- języki programowania.

System operacyjny

System operacyjny jest to specjalny program do kontroli i koordynacji działania całego sprzętu. Nadzoruje on wprowadzanie i wyprowadzanie danych, obliczenia i zapamiętywanie informacji tak, aby działania odbywały się w sposób poprawny i dobrze zorganizowany. W wielu komputerach osobistych system operacyjny jest stosunkowo prostą częścią oprogramowania mającą ograniczone możliwości i zapewniającą najczęściej jedynie proste funkcje np. wydruk, zapamiętanie, wywołanie i uruchomienie programu. W przypadku komputerów profesjonalnych (szczególnie tych, które mogą być wykorzystywane przez wielu użytkowników), system operacyjny jest to niesłychanie skomplikowany, a często także największy program jaki pracuje w komputerze.

Języki programowania

Język programowania pozwala na pisanie programów względnie szybko i w postaci zbliżonej bardziej do normalnego zapisu języka ludzkiego niż do abstrakcyjnego, wewnętrznego kodu komputera. Wiele komputerów osobistych ma tylko jeden język programowania, a mianowicie Basic. Komputery profesjonalne pozwalają na posługiwanie się kilkoma językami, spośród których Fortran, Pascal i C są najbardziej popularne w obliczeniach naukowych i w grafice komputerowej.

Ograniczenia standardowych języków programowania

Opisane elementy sprzętu i oprogramowania są zwykle dostarczane przez producenta jako uniwersalny zestaw graficzny, umożliwiający pisanie programów użytkowych dla własnych potrzeb. Ponieważ żaden z wyżej wymienionych języków programowania nie był specjalnie przewidziany do prac związanych z grafiką, producenci często dostarczają rozszerzone wersje języków zawierające odpowiednie instrukcje graficzne takie jak DRAW, MOVETO, CIRCLE itp.

Producenci wykorzystują specjalne cechy własnych produktów i wobec tego w zasadzie nie ma żadnej standaryzacji tych rozszerzeń. Jest więc regułą, że program graficzny napisany dla jednego typu komputera nie jest w stanie pracować z innym, bez modyfikacji. Ten brak standaryzacji utrudnia przenoszenie oprogramowania. Stwarza także problemy przy pisaniu książek takich jak ta, w których - aby nie być zbyt abstrakcyjnym - należy używać instrukcji zorientowanych na określony komputer i język programowania. Wyraźnie widać pilną potrzebę podjęcia prób standaryzacji w przemyśle komputerowym - przynajmniej w zakresie możliwości graficznych. Wymaga to jednak czasu, nim wpływ standaryzacji ujawni się w produkcji komputerów i języków programowania. W książce starano się przewidzieć akceptację standardów graficznych przez podanie ogólnych przykładów programowych, uwzględniających obecne tendencje.

Wszystkie elementy w systemie są ważne i defekty w którymkolwiek z nich ograniczają efektywność całości. Jakość i typy rysunków graficznych oraz sposób konwersacji z użytkownikiem zależą od dostępnego sprzętu i oprogramowania. Osiągnięte wyniki przy pełnym wykorzystaniu możliwości systemu zależą od jakości i zrozumienia dokumentacji.

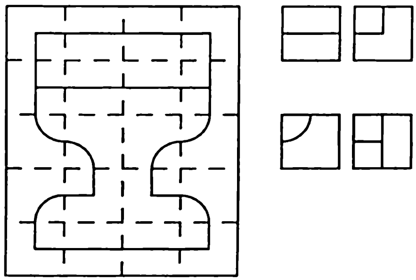
2.3. Style graficzne

Różne systemy graficzne, a zwłaszcza dostępne urządzenia wyjściowe realizują trzy style grafiki komputerowej. Są to:

- grafika znakowa lub mozaikowa,
- grafika kaligraficzna lub wektorowa,
- grafika rastrowa lub pikselowa.

Grafika znakowa lub mozaikowa

Rysunek 2.4 został zestawiony - podobnie jak zestawia się mozaiki - ze specjalnych znaków graficznych programu dostarczanego wraz z komputerem lub zaprojektowanych przez użytkownika. Znaki te wprowadza się z klawiatury, tak jak zwykłe znaki al-



Rys.2.4. Rysunek typu mozaikowego i jego elementy składowe

fanumeryczne. Przy starannym projekcie i dużej pomysłowości można osiągnąć całkiem intresujące wyniki, ale rysowanie za pomocą znaków ma bardzo ograniczone możliwości i nadaje się tylko do tworzenia prostych diagramów, wideotekstów i rysunków do gier. Niektóre z komputerów osobistych posługują się tylko taką grafiką.

Grafika kaligraficzna lub wektorowa

Ten styl rysowania pozwala na tworzenie najdokładniejszych rysunków na papierze i jest szczególnie przydatny w zastosowaniach inżynierskich i kreślarskich. W tym przypadku rysunki są złożone z linii (na ogół w jednym kolorze). Za pomocą takiej grafiki trudno jest uzyskać obszary zamalowane jednym kolorem, dlatego nie stosuje się jej wówczas, gdy są potrzebne rysunki cieniowane lub dobry realizm. Ponieważ monitory do grafiki wektorowej są drogie, nie jest ona stosowana w komputerach osobistych.

Grafika rastrowa lub pikselowa

Ten rodzaj grafiki umożliwia tworzenie rysunków złożonych ze zbiorów kropek położonych obok siebie. Kropki zwane pikselami (ang. *picture elements*) pozwalają na wykreślanie linii lub obszarów w różnych kolorach. Grafika rastrowa weszła do powszechnego użytku ponieważ pozwala na zastosowanie prostszego sprzętu. Jednakże kropkowa faktura rysunków stwarza problemy z rozdzielczością i dokładnością, które muszą być rozwiązane metodami programowymi, a czasem także i sprzętowymi. Przykładem powstających problemów jest wygląd linii ukośnych, mających denerwującą "schodkową" strukturę. Ten sposób rysowania jest stosowany w większości komputerów osobistych. W książce zostanie omówiona jedynie grafika rastrowa i wektorowa.

2.4. Zadania graficzne

Zadania systemów graficznych ułatwiają wprowadzanie i wyprowadzanie informacji graficznych. Zadania do wprowadzania informacji:

- konwersacja z systemem,
- definiowanie i redakcja rysunków i tekstów graficznych,
- przekształcanie istniejących rysunków do postaci akceptowalnej przez komputer; proces ten nosi nazwę dyskretyzacji.

Zadania do wyprowadzania informacji:

- wykonywanie rysunków,
- tworzenie menu graficznych lub symboli ułatwiających konwersację z systemem.

Komputery graficzne różnią się od komputerów ogólnego przeznaczenia tym, że zawierają inne urządzenia wejściowe i wyjściowe wraz ze wspomagającym je oprogramowaniem. Natomiast drogie komputery są wyposażone w specjalne procesory do prac graficznych. Dla większości zastosowań obydwa rodzaje urządzeń są równie ważne. Sprzedawcy systemów graficznych starają się podkreślić jakość produktu (parametry) urządzeń wyjściowych. Oczywiście, że imponujące wyniki mogą być pokazane tylko wówczas, gdy system jest w stanie je wyprodukować. Użytkownik szybko wykryje, że dobrze zaprojektowane i oprogramowane zadanie wprowadzania informacji jest ważne, jeśli właściwości systemu mają być wykorzystane w pełni.

Instrukcje graficzne

Do realizacji zadań graficznych systemy są wyposażone w specyficzne dla nich zestawy instrukcji graficznych, zwanych inaczej *prymitywami graficznymi*. Niektóre systemy, zwłaszcza te droższe, mają obszerny zbiór prymitywów do rysowania takich elementów jak linie, wielokąty, okręgi, łuki i inne figury; do zmiany postaci linii; do wypełniania lub kreskowania obszarów; do wymazywania lub wyświetlania części rysunku; do obracania, przesuwania lub tworzenia odbicia lustrzanego rysunku itp. Wszystkie te działania są wykonywane za pomocą pojedynczych instrukcji. Inne systemy mają bardziej ograniczony zestaw prymitywów, często pozwalających jedynie na wyświetlenie linii w jednej postaci i w jednym kolorze. Systemy różnią się nie tylko swoimi

możliwościami wyprowadzania informacji, ale także wprowadzania danych; niektóre nie mają żadnych prymitywów wejściowych, inne zapewniają zbieranie danych z wielu różnorodnych źródeł wejściowych.

W celu zrozumienia przykładów założono, że system ma tylko sześć prymitywów: jeden do informowania systemu o chęci korzystania z jego możliwości graficznych, cztery do wyprowadzania i jeden do wprowadzania informacji. Przyjęto, że wartości (x, y) współrzędnych podawanych w prymitywie leżą w zakresie dopuszczonym przez powierzchnię, na której jest wykonywany rysunek (rys. 2.5).
Wartości poza tym zakresem spowodują powstanie błędu. Oto założone prymitywy:

GRAPHICS(n)
Instrukcja ta wymazuje ekran i ustawia system w tryb graficzny wówczas, gdy n = 1 lub w tryb tekstowy (bez wymazywania ekranu) jeśli n = 0. Wszystkie instrukcje graficzne użyte wtedy, kiedy n = 0 (albo jeśli n nie jest zdefiniowane) są ignorowane.

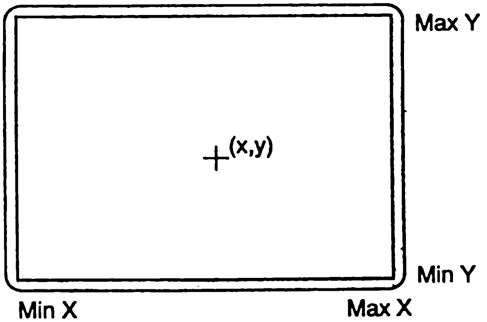
MOVE(x, y)
Instrukcja ta umieszcza głowicę kreślącą (strumień elektronów lub pisak) w pozycji określonej przez współrzędne (x, y), bez kreślenia czegokolwiek.

DRAW(x, y)
Instrukcja ta przesuwa głowicę kreślącą z obecnego położenia do nowego położenia (x, y), pozostawiając ślad przejścia w postaci linii prostej.

TEXT(x, y, komunikat)
Instrukcja ta drukuje ciąg znaków określony przez komunikat poczynając od miejsca o współrzędnych (x, y).

COLOUR(n)
Instrukcja ta ustawia kolor rysowanych, kolejnych elementów. Gdy n = 0, to będą rysowane elementy w kolorze tła, co oznacza, że narysowany element będzie niewidoczny. Obiekty o kolorze COLOUR(0) pozwalają na wymazywanie istniejących już elementów przez zamalowanie ich kolorem tła.

ACCEPT(znacznik, x, y)
Instrukcja ta pobiera parę współrzędnych (x, y) z urządzenia wejściowego i ustawia znacznik logiczny na wartość zależną od akcji podjętej przez użytkownika (takiej jak na przykład naciśnięcie przycisku).

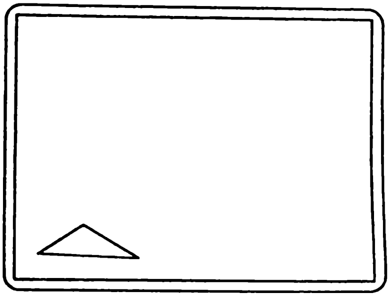


Rys. 2.5. Powierzchnia rysunkowa (wszystkie punkty muszą leżeć wewnątrz tj. między Min X a Max X w poziomie i między Min Y a Max Y w pionie)

Zatem w systemie o tak założonych prymitywach, w celu narysowania trójkąta na ekranie poczynając od punktu (15, 30), o pozostałych wierzchołkach w punktach (120, 17) i (60, 50), należy użyć instrukcji:

```
GRAPHICS(1)
COLOUR(1)
MOVE(15, 20)
DRAW(120, 17)
DRAW(60, 50)
DRAW(15, 20)
```

W efekcie powstanie trójkąt pokazany na rys. 2.6.



Rys. 2.6. Trójkąt na ekranie

Jest mało prawdopodobne, że system używany przez Czytelnika będzie miał dokładnie ten sam zestaw prymitywów jak podany. Będzie miał jednak ich odpowiedniki, należy zatem podstawić instrukcje dostępne w jego systemie przed użyciem jakichkolwiek podprogramów podanych w tej książce. Na przykład użytkownicy komputera Apple II będą musieli przy korzystaniu z Basicu napisać program w postaci:

```
100 HGR
110 HCOLOR = 1
120 HPLOT 15,20 TO 120,17 TO 60,50 TO 15,20.
```

W rzeczywistości, wobec odmienności sposobu definiowania współrzędnych przez komputer Apple II, instrukcje te spowodują narysowanie trójkąta odwróconego w stosunku do pokazanego na rysunku. Komputer Apple II może rysować linie w 7 kolorach, a więc parametr instrukcji HCOLOR może przyjmować wartości z zakresu od 1 do 7. Korzystając z zestawu przyjętych prymitywów, w książce będą pokazane możliwości zbudowania zestawu instrukcji, przypominającego zestawy spotykane w profesjonalnym sprzęcie graficznym.

ZADANIA

2.1. Sporządź listę prymitywów graficznych swojego systemu komputerowego i porównaj ją z podaną w książce. Lista będzie prawdopodobnie większa niż tu

podana, ale prosimy wskazać instrukcje równoważne podanym. Zachowaj tę listę na przyszłość, gdyż będzie potrzebna do pisania programów.

2.2. Używając tych prymitywów napisz program do rysowania trójkąta i porównaj wynik z rys. 2.6. Czy figura może być poprawnie rozpoznana jako trójkąt, czy też jest zbyt mała, aby mogła być dobrze narysowana? Spróbuj wykonać zadanie przy różnych wartościach liczb oraz zobacz, co się stanie, jeśli użyjesz liczb rzeczywistych zamiast liczb całkowitych.

2.3. Czy początek układu współrzędnych w Twoim komputerze znajduje się w lewym, dolnym rogu czy też w innym miejscu? Jeżeli znajduje się w lewym, górnym rogu, to co zrobić aby rysunki nie były odwrócone ("do góry nogami")? Wskazówka: Co się zdarzy jeśli odejmiesz wszystkie wymiary pionowe rysunku od maksymalnego wymiaru ekranu?

2.4. Co nastąpi jeśli zażądasz od systemu narysowania linii ukośnej, której koniec znajduje się poza obrębem ekranu? Czy system zaprotestuje w jakiś sposób, zniszczy rysowaną linię czy też narysuje tyle, na ile pozwala obszar do rysowania? Proszę eksperymentować przy różnych warunkach, gdyż wynik tego eksperymentu jest ważny.

3. Zadania i urządzenia wejściowe

3.1. Zadania wejściowe

Cechy, jakie powinny mieć urządzenia wejściowe zależą w znacznym stopniu od zastosowanego rodzaju *grafiki komputerowej: konwersacyjnej* lub *niekonwersacyjnej*. W przypadku grafiki konwersacyjnej zadania wejściowe są wykonywane na bieżąco z wizualnym sprzężeniem zwrotnym, natomiast w grafice niekonwersacyjnej - zadania wejściowe są wykonywane jako niezależny proces, oddzielony od wyjścia. Trzy podstawowe zadania urządzeń wejściowych to konwersacja z użytkownikiem, tworzenie rysunków i przekształcanie na postać cyfrową; w przypadku grafiki niekonwersacyjnej zostaje pominięte pierwsze zadanie (łącznie z niezbędnym sprzętem). W obydwu przypadkach, podstawowym zadaniem graficznych urządzeń wejściowych jest przyjęcie informacji i przesłanie jej do procesora. W przypadku grafiki niekonwersacyjnej informacja ta przedstawia sobą opis rysunku (lub inaczej opis obiektu) do wyświetlenia. W przypadku grafiki konwersacyjnej, informacja ta może przedstawiać sobą nie tylko obiekty graficzne, lecz także dane, których dokładne właściwości procesor będzie interpretował w zależności od kontekstu. Po odebraniu informacji procesor wyświetla wynik na urządzeniu wyjściowym; użytkownik otrzymuje wizualne potwierdzenie, że jego instrukcje zostały poprawnie wykonane.

Konwersację można ułatwić przez użycie *menu*, stanowiącego listę elementów. Menu pojawia się w obszarze wydzielonym, czasowo lub na stałe, na ekranie lub innym urządzeniu wejściowym. Listę mogą stanowić instrukcje takie jak DRAW, MOVE, ROTATE itp., albo obiekty graficzne takie jak: symbole przedstawiające stoły, krzesła lub symbole elementów elektronicznych. W niektórych komputerach, np. Apple Lisa i Macintosh, symbole w menu są obrazkowe i przedstawiają raczej zadania lub koncepcje niż obiekty rzeczywiste. Użyte w takim kontekście symbole są znane pod nazwą *ikon*. Typowymi ikonami są: kosz na śmieci do przedstawienia zadania usuwania pewnej pracy, kartoteka do wyrażenia zapamiętywania informacji lub słoik kleju do przedstawienia klejenia jednego rysunku na drugim. Jeżeli użytkownik wybierze obiekt z menu, to procesor wykona instrukcję lub umożliwi pracę z wybranym obiektem.

3.2. Graficzne urządzenia wejściowe

Przemysł komputerowy już dawno docenił różnorodność i złożoność zadań wejściowych grafiki i oprócz klawiatury produkuje dużo różnych urządzeń wspomagających te prace. Do urządzeń wejściowych należą:

- pióra świetlne,
- myszy,
- manipulatory (joysticki, kule do śledzenia i pokrętła),
- panele i płytki dotykowe,
- gałki, skale i suwaki,
- przyciski i klawisze funkcyjne,
- tablice graficzne i dyskretyzatory,
- skanery i urządzenia do śledzenia linii.

Urządzenia różnią się sposobami użycia, dostępnością i ceną. Niektóre z nich dobrze nadają się do jednego lub dwóch z podstawowych zadań wejściowych, ale tylko nieliczne są w stanie realizować wszystkie trzy zadania. Do konwersacji, tworzenia rysunków i przekształcania na postać cyfrową urządzenie musi spełniać funkcje sześciu "umownych" lub logicznych urządzeń, zależnie rodzaju wykonywanej pracy. Tymi logicznymi urządzeniami są:

- lokator informujący procesor o położeniu na powierzchni roboczej,
- wskaźnik identyfikujący obiekt,
- przycisk lub wybierak służący do wybrania opcji, ze zbioru
- waluator podający do procesora takie wielkości jak kąt lub odległość,
- łańcuch lub klawiatura podająca do procesora ciąg znaków (tytuły lub inne opisy),
- wektor podający do procesora położenie pozycji kursora zmieniającej się przy pisaniu lub rysowaniu.

Możliwe jest użycie konwencjonalnej klawiatury do symulacji działania niektórych z tych urządzeń logicznych np. przez wpisywanie położeń, wartości, instrukcji itp.

Przykład

```
DRAW FROM (10,12) TO (100,100)
MOVE BOX-1 TO (30,60)
ROTATE OBJECT-6 BY 45 DEGREES
COLOUR = RED
X = 20
```
◆

Mimo, że metoda ta ma zalety przy pewnych pracach, nie zawsze jest wystarczająca i do bardziej złożonych zadań należy stosować lepsze urządzenia.

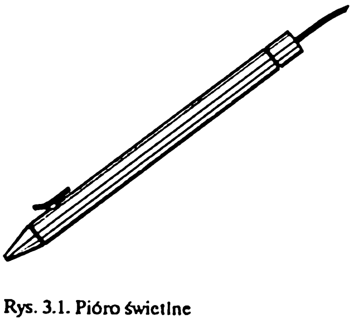
Do prawidłowego działania, wiele urządzeń wymaga aby procesor wyświetlał stale na ekranie kursor graficzny, zwykle w postaci krzyża pokazującego aktualne położenie systemu graficznego. Przesuwanie elementu wskazującego w urządzeniu wejściowym powoduje odpowiednie przesunięcie krzyża. Zwykle użytkownik może wybrać określone położenie kursora naciskając przycisk lub inicjując podobną akcję.

Pióro świetlne

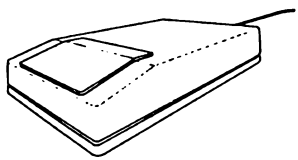
Pióro świetlne (rys. 3.1) jest to urządzenie o wymiarach grubego długopisu, dołączonego do komputera za pomocą cienkiego kabla. W przypadku naciśnięcia przycisku znajdującego się na piórze i dotknięcia piórem powierzchni ekranu, procesor wykrywa ten fakt i określa pozycję wskazywanego obiektu. Przy użyciu odpowiedniego oprogramowania, pióro świetlne znakomicie nadaje się jako lokator lub wskaźnik do wybierania obiektów znajdujących się na ekranie. W tym zastosowaniu pióro świetlne jest wygodnym urządzeniem, atrakcyjnym dla wielu ludzi. Pióro świetlne nie nadaje się jednak do wszystkich rodzajów ekranów i ma ograniczoną dokładność. Pracuje ono na zasadzie detekcji światła z ekranu i nie jest w stanie odróżnić dwóch punktów, które są położone zbyt blisko siebie, także odbicia z ekranu traktuje jako rzeczywiste obiekty. Chociaż za pomocą pióra świetlnego można w ograniczonym zakresie wykonywać rysunki, to jednak stosuje się je głównie do konwersacji. Stosując dodatkowo specjalne metody programowe, można uczynić je dokładnym waluatorem.

Mysz

Mysz (rys. 3.2) jest to urządzenie elektromechaniczne w kształcie poręcznego pudełka dołączonego do komputera za pomocą kabla. Zaletą tego urządzenia jest możliwość użycia na dowolnie płaskiej powierzchni, np. na biurku. Mysz jest zmontowana na dwóch kółkach, ustawionych pod kątem prostym, wzdłuż osi głównych. Ten rodzaj myszy jest jednym z wielu urządzeń wejściowych wykorzystujących potencjometry do obliczania położenia. Kółka myszy są przymocowane do osi dwóch potencjometrów, których rezystancja jest proporcjonalna do ruchu kół. Mysz, w miarę manewrowania nią, przesyła aktualną wartość rezystancji. W myszy, w której zamiast kół jest zainstalowana kulka, wielkość przesunięcia jest rozpoznawana przez czujniki elektrooptyczne. Analogowe sygnały przychodzące z myszy zostają zamienione za pomocą przetwornika analogowo-cyfrowego na sygnały cyfrowe akceptowane przez komputer. Informacje przesłane do komputera służą do obliczania nowego położenia w stosunku do poprzednio ustalonego; aktualne położenie wskazywane jest na ekranie kursorem np. w postaci krzyża.



Rys. 3.1. Pióro świetlne



Rys. 3.2. Mysz

Przy ruchu myszy zawsze występuje pewien poślizg i jest również możliwe przeniesienie myszy bez przesuwania jej kół. Dlatego też mysz nie jest szczególnie dokładnym urządzeniem wejściowym a może być wręcz trudna w użyciu przy tworzeniu skomplikowanych rysunków. Przy pewnej praktyce można jej skutecznie używać np. do wykonywania szkiców. Doświadczenie pokazuje, że jest ona szczególnie wygodna jako urządzenie do wybierania, wskazujące wyświetlone elementy oraz do przenoszenia ich po ekranie w trybie redagowania. Mysz ma dużą liczbę entuzjastycznych zwolenników i coraz bardziej staje się preferowanym urządzeniem do konwersacji w nowych systemach graficznych.

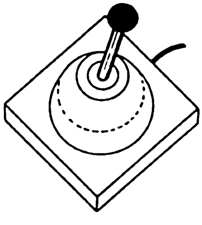
Manipulatory (joysticki, kule śledzące, pokrętła)

Manipulatory należą do grupy pokrewnych urządzeń, które podobnie jak mysz wskazują położenie, podawane w postaci pary współrzędnych (X,Y), zależnych od przekazywanej zmiany wartości rezystancji. Aktualne położenie jest pokazywane na ekranie za pomocą krzyża lub innego kursora graficznego.

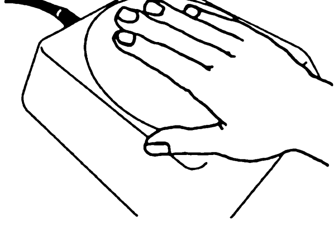
Joystick (rys. 3.3) (wolant - polska nazwa, przyjęta z uwagi na podobieństwo do wolantu w samolocie), jest krótkim drążkiem obracającym się na przegubie kulistym, co pozwala na swobodny ruch w płaszczyźnie poziomej. Konstrukcja joysticku zapewnia, że ruchy boczne powodują zmianę współrzędnej X, a ruchy w przód i w tył - współrzędnej Y. Zmiany obydwu współrzędnych następują w wyniku ruchu ukośnego. Niektóre joysticki są tak zbudowane, że sprężyna przywraca środkowe położenie drążka po ruchu. Inne przekazują informację nieliniową; im bardziej drążek jest odchylony od położenia środkowego, tym szybszy jest ruch kursora. Prosty joystick jest stosunkowo tani, dostosowany do większości komputerów i jest dobrym lokatorem.

Kula śledząca (rys. 3.4) ma konstrukcję wykorzystującą ruch kuli w gnieździe, która poruszana palcami lub dłonią obraca się swobodnie. Urządzenie jest dość duże; kula ma średnicę od 100 do 150 mm i jest wskaźnikiem bardzo elastycznym. Nie nadaje się jednak do wykonywania dużych ruchów. Kula śledząca jest droga i dlatego jest stosowana tylko w nielicznych systemach.

Chociaż joystick i kula śledząca są łatwe w użyciu i mogą być stosowane bezpośrednio do rysowania, to najlepiej spełniają funkcję śledzenia i wskazywania obiektów na ekranie. Ich duża mobilność czyni je uprzywilejowanymi urządzeniami w zasto-



Rys. 3.3. Manipulator (joystick)



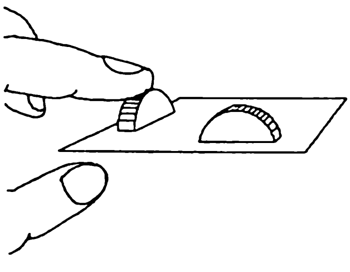
Rys. 3.4. Kula śledząca

sowaniu do gier komputerowych. Mają również poważne zastosowania, np. do wskazywania obiektów na ekranie radarowym, gdzie odpowiedź musi być bardzo szybka.

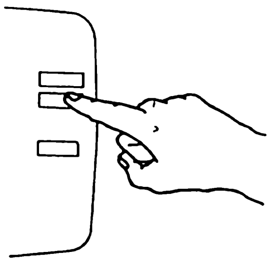
Pokrętła (rys. 3.5) są to dwa kółka z moletowanym brzegiem, ustawione pod kątem prostym w stosunku do siebie i wbudowane w klawiaturę komputera. Z uwagi na ich bliskie rozmieszczenie można manipulować jednocześnie za pomocą jednej ręki i niezależnie ustawić współrzędne X i Y, co jest dużą zaletą. W urządzeniach graficznych firmy Tektronix, współrzędne są przedstawiane na ekranie jako oddzielne linie poziome i pionowe (nie jako krzyż) i dlatego jest łatwo tworzyć dokładne rysunki złożone z linii poziomych i pionowych, równoległych do brzegów ekranu. Takiego zadania nie daje się w prosty sposób zrealizować za pomocą wolantu lub myszy.

Panele i płytki dotykowe

Panel dotykowy (rys. 3.6) jest urządzeniem bardziej bezpośrednim w użyciu niż pióro świetlne, gdyż wymaga jedynie dotknięcia ekranu palcem. Taka konwersacja nie pozwala z natury rzeczy na osiągnięcie dużej dokładności; typowa rozdzielczość wynosi 50 na 50 punktów. Urządzenie jest więc przydatne do wybierania obiektów z menu na ekranie lub podobnych zadań. Prostota i bezpośredniość czyni je szczególnie użytecznym dla nie przeszkolonych osób. Panel dotykowy wymaga specjalnego ekranu lub nakładki na ekran, wykorzystujących metody elektromagnetyczne, elektrooptyczne lub akustyczne do określania dystansu.



Rys. 3.5. Pokrętła



Rys. 3.6. Panel dotykowy

Płytki dotykowe są odmianą paneli dotykowych i składają się z prostokątnego kawałka gumy lub plastyku umieszczonego na podłożu czułym na nacisk. Dotyk lub ruch palca na płytce powoduje wysłanie odpowiedniego sygnału do komputera, a zatem przesunięcia kursora na ekranie. Te urządzenia nie są stosowane we wszystkich systemach graficznych i nadają się jedynie do konwersacji.

Gałki, skale i suwaki

Gałki, skale i suwaki chociaż rzadko używane jako wejściowe urządzenia graficzne są stosowane w niektórych systemach, najczęściej do określania kąta widzenia obiektów trójwymiarowych. Podobnie jak w joysticku i kuli śledzącej, sygnał jest otrzymywany z potencjometrów i wymaga przetworzenia analogowo-cyfrowego.

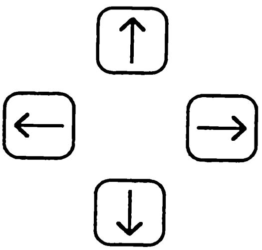
Przyzwyczajenie do używania gałek i suwaków w celu ustawiania parametrów sprawia, że niektórzy użytkownicy piór świetlnych tworzą dla tych urządzeń programowe opcje na ekranie. Przesuwanie pióra po okręgu lub wzdłuż skali daje taki sam wynik jak manipulacja gałkami lub suwakami. Przez odpowiednie skalowanie można osiągnąć znaczną dokładność tych programowych waluatorów, wybieranych dowolnym urządzeniem wejściowym.

Innym typem urządzenia do ustawiania potencjometru są drążki (ang. *paddles)* dostarczane parami wraz z komputerami osobistymi. Drążek to zwykle para potencjometrów w plastykowej obudowie, połączonych z procesorem za pomocą cienkiego kabla. Mimo, że drążki nie są tak wygodne jak pokrętła, to są w stanie spełniać takie same funkcje. Nie są jednak tak dokładne.

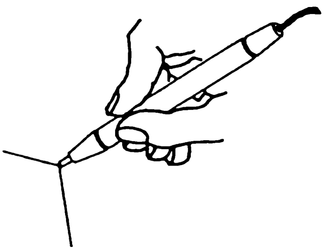
Przyciski i klawisze funkcyjne

Niektóre z systemów graficznych są wyposażone w zestawy klawiszy lub przycisków funkcyjnych, zarezerwowanych do specjalnych celów. Jeden z takich zestawów może służyć do kontroli ruchu kursora po ekranie. Naciskanie klawiszy, często zgrupowanych we wzór mnemoniczny (rys. 3.7), powoduje przesuwanie kursora po ekranie w pożądanym kierunku. Chociaż nie jest to szczególnie wygodna metoda działania, to jednak umożliwia rysowanie prostych rysunków. Niezależność ruchów kursora wywoływanych klawiszami - podobnie jak w przypadku pokręteł - ułatwia tworzenie rysunków złożonych z linii prostych, równoległych do brzegów ekranu.

W niektórych przypadkach systemy graficzne są wyposażone w zestawy specjalnych klawiszy, przeznaczonych do różnych prac graficznych. Funkcje klawiszy mogą być predefiniowane przez producenta lub określane przez program użytkowy, zgodnie z którym można za pomocą plastykowych nakładek na te klawisze, opisać ich funkcje. Naciśnięcie klawisza jest równoznaczne z wyborem elementu z menu. Jeżeli nawet



Rys. 3.7. Klawisze do kontroli ruchu kursora



Rys. 3.8. Rysik

producent nie przewidział takiej możliwości, to użytkownik może łatwo napisać program pozwalający na użycie klawiszy konwencjonalnej klawiatury do realizacji podobnych funkcji. Posiadanie specjalnych zestawów klawiszy jest jednak wygodniejsze.

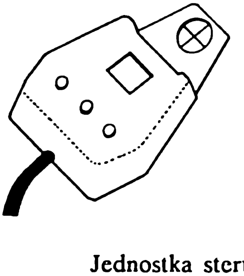
Tablice graficzne i dyskretyzatory

Tablica graficzna lub dyskretyzator składa się z trzech połączonych części:

- cienkiej, płaskiej płyty (zwanej tablicą graficzną), tworzącej powierzchnię roboczą;
- urządzenia wskazującego, które może być przemieszczane po płycie;
- jednostki sterującej, przekształcającej sygnały elektryczne, powstające w wyniku zadziałania wskaźnika i płyty po otrzymaniu informacji o położeniu odniesionym do początku układu współrzędnych. Sygnał o działaniu wskaźnika powstaje zwykle w wyniku indukcji elektromagnetycznej, czasem w wyniku zmiany rezystancji płyty, albo - rzadko - za pomocą akustycznych metod określania odległości.

Płyty są produkowane w różnych rozmiarach, od 11" (około 280 mm) do 60" (około 1500 mm), przy czym płyty o większych rozmiarach mają stojak. W niektórych wersjach, płyta jest przezroczysta, co pozwala na podświetlanie jej od tyłu dla ułatwienia dyskretyzacji klisz rentgenowskich lub przezroczy.

Elementem wskazującym jest rysik (rys. 3.8) lub celownik (rys. 3.9). Rysik, kształtem podobny do pióra, przez włączenie przycisku i dociśnięcie końcówki do płyty uaktywnienia się. Celownik zawiera szklane okienko z krzyżowym celownikiem (stąd nazwa) i obudowę z kilkoma przyciskami funkcyjnymi, których naciśnięcie wywołuje określone działania. W większości przypadków obydwa te urządzenia są zainstalowane, wybór jednego z nich jest kwestią indywidualnych upodobań użytkownika. Celownik lepiej nadaje się do dyskretyzacji rysunków, a rysik do wskazywania i wybierania.



Rys. 3.9. Celownik

Jednostka sterująca działa jako bufor między wskaźnikiem i płytą a komputerem w celu zapewnienia ich poprawnej współpracy. W niektórych wersjach jednostka ma pamięć buforową o znacznej pojemności, co umożliwia zapamiętanie pewnej liczby współrzędnych przed wysłaniem ich do procesora. Tablica graficzna i dyskretyzator mogą pracować w *trybie punktowym* lub w *trybie strumieniowym*. O wyborze trybu pracy decyduje jednostka sterująca.

W trybie strumieniowym, do procesora jest wysyłany z dużą prędkością, ciągły strumień informacji o współrzędnych punktów. Szybkość przesyłania jest regulowana, co pozwala na gęste lub rzadkie rozmieszczenie punktów w miarę ruchu celownika po płycie. Ten rodzaj pracy nadaje się najlepiej do wprowadzania krzywych i linii niere-

gularnych, ale przy większych prędkościach wejściowych powoduje szybkie zapełnienie całej dostępnej pamięci. Strumieniowy tryb pracy może programowo wykrywać ruch wskaźnika przez proste porównanie współrzędnych w danym momencie ze współrzędnymi poprzednimi. W tym trybie pracy urządzenie może być traktowane jako urządzenie wektorowe. Podczas pracy w trybie punktowym, współrzędne są przesyłane tylko wówczas, gdy naciśnięty jest odpowiedni klawisz funkcyjny lub koniec rysika. Ten rodzaj pracy ma zastosowanie do wybierania takich elementów jak końce linii.

Niektóre jednostki sterujące są wyposażone we wskaźniki zbudowane z diod świecących (LED), pokazujące współrzędne X i Y wybranej pozycji. Zależnie od rodzaju jednostki sterującej (jej oprogramowania) istnieje możliwość przesunięcia rysunku na płycie i obrócenia tak, aby osie rysunku były równoległe do osi płyty. Jest to ważna funkcja, którą musiałby zrealizować program użytkownika.

Staranne używanie dyskretyzatorów i tablic graficznych umożliwia osiągnięcie dużej dokładności otrzymanego obrazu. Niektóre z profesjonalnych modeli tych urządzeń mają rozdzielczość do 0.025 mm lub większą. Tablice graficzne sprzedawane do komputerów osobistych mają rozdzielczość do 0.4 mm. Uzyskiwana dokładność powoduje, że dyskretyzatory są powszechnie używane do wykonywania - za pomocą komputera - map, rysunków technicznych i projektów architektonicznych.

Opisany dyskretyzator jest dwuwymiarowy, przeznaczony do przekształcania rysunków dwuwymiarowych na zbiór współrzędnych w celu dalszej obróbki i wyświetlania. Niejednokrotnie istnieje konieczność dyskretyzacji rysunków obiektów trójwymiarowych, które są trudniejsze do przetworzenia na postać cyfrową - zwłaszcza jeśli zawierają powierzchnie krzywoliniowe. Dostępne są nieliczne dyskretyzatory trójwymiarowe i są traktowane jako egzemplarze eksperymentalne.

Skanery i urządzenia do śledzenia linii

Skanery i urządzenia do śledzenia linii stanowią grupę automatycznych i półautomatycznych dyskretyzatorów, przeznaczonych do przetwarzania istniejących rysunków lub fotografii na postać akceptowalną przez komputer. Skanery są wyposażone w kamerę telewizyjną (wtedy są zwane przetwornikami ramki) lub są wykonywane jako skanery przyrostowe Działanie pierwszego skanera polega na ustawieniu kamery na żądany obiekt przeznaczony do wprowadzenia do komputera. Sprzęt i program przetwarza sygnał wizyjny obrazu na postać cyfrową. Obraz jest zapisywany do pamięci jako wzór pikseli w celu późniejszej obróbki. Zazwyczaj skanery telewizyjne przetwarzają tylko obraz czarno-biały o pewnej liczbie poziomów szarości. Urządzenia te są stosunkowo tanie i mogą stanowić wyposażenie komputerów osobistych. Rozwój półprzewodnikowych czujników elektrooptycznych, takich jak matryce ze sprzężeniem ładunkowym (CCD) i optyczne pamięci o dostępie przypadkowym, mające bezpośredni dostęp do sygnałów wizyjnych, przyczyni się zapewne do rozpowszechnienia skanerów telewizyjnych w przyszłości.

Skanery przyrostowe są to urządzenia elektromechaniczne, w których głowica z fotoelementem przesuwa się po rysunku lub fotografii według ustalonego rastra i przekształca poziom jasności obiektu na postać cyfrową. Urządzenia te są raczej drogie i trudno dostępne. Skanery przyrostowe są używane tylko do prac badawczych oraz

do wprowadzania zdjęć o wysokiej jakości do publikacji. Umożliwiają również łatwe i szybkie wprowadzanie kolorowych zdjęć.

Informacje graficzne wprowadzane do komputera za pomocą skanerów mają charakter fotografii, a przetwarzanie ogranicza się do wykonania prac podobnych do jej obróbki, a zatem: powiększania, odwrócenia stronami, skręcania, zwijania, uzyskania negatywu itp. Na obecnym etapie rozwoju oprogramowania nie jest możliwe wymaganie od komputera zrozumienia struktury obrazu. Tak więc jeśli wprowadzimy dowolną metodą rysunek architektoniczny, to nie możemy zażądać od systemu, aby rozjaśnił zewnętrzne ściany lub usunął wszystkie wewnętrzne drzwi itd.; system nie ma dostatecznej wiedzy na temat rysunku, aby był w stanie wykonać tę pracę.

Urządzenia do śledzenia linii pozwalają na wprowadzanie w sposób półautomatyczny informacji o strukturze dobrze określonej. Urządzenia te składają się z płaskiego podłoża, na którym porusza się głowica śledząca, zamocowana do przegubowego ramienia. Głowicę ustawia się ręcznie nad początkiem wprowadzanej linii i wprawia się ją w ruch. Urządzenie śledzi automatycznie linię zgodnie z jej przebiegiem i wysyła do procesora sygnały tak długo aż dojdzie do miejsca gdzie linia rozgałęzia się lub ma skrzyżowanie z inną linią (lub powróci do miejsca startu). Tam oczekuje na podanie następnej drogi. Przez zadanie dróg do śledzenia, w dobrze uporządkowanej kolejności można przesłać do komputera informacje o strukturze rysunku. Przykładowy rysunek architektoniczny można wprowadzić do komputera w następującej kolejności: najpierw należy zadać śledzenie ścian zewnętrznych, potem wewnętrznych, potem drzwi, mebli itd. W ten sposób zapamiętane elementy pozwalają na łatwą ich identyfikację przy dalszych pracach. Urządzenia do śledzenia linii są drogie i trudno dostępne.

Urządzenia wejściowe - podsumowanie

Dobry system graficzny, oprócz konwencjonalnej klawiatury alfanumerycznej powinien mieć dołączone jedno z opisanych urządzeń wejściowych. Większość wytwórców zapewnia możliwość dołączenia takich urządzeń do komputera. Należy stwierdzić, że nie da się wykonać żadnej poważnej pracy graficznej bez pomocy jakiegoś dyskretyzatora lub tablicy graficznej. Użytkownicy o mniejszych zasobach finansowych powinni rozważyć możliwość zakupu tańszego modelu tych urządzeń zamiast innych urządzeń wejściowych.

Wprowadzanie informacji jest jedną z głównych prac w zakresie grafiki komputerowej. W wielu jej praktycznych zastosowaniach, takich jak projektowanie wspomagane komputerem lub animacja komputerowa, wprowdzanie danych w dobrze określonej formie jest jedną z najbardziej czasochłonnych i najtrudniejszych prac. Bez odpowiednich narzędzi sprzętowych i programowych praca ta staje się niesłychanie żmudna i nie jest pozbawiona błędów.

4. Zadania i urządzenia wyjściowe

4.1. Zadania wyjściowe

System graficzny ma dwa zadania. Pierwsze, najbardziej oczywiste, to wytwarzanie rysunków takich jak: mapy, szkice techniczne, obrazy o mniejszym lub większym realiźmie, diagramy i wykresy do celów handlowych i naukowych, a nawet filmy animowane. Działania te możemy określić wspólnym mianem *prac rysunkowych*. Drugie (dla zaczynających pracę z grafiką komputerową wcale nie jest to zadanie oczywiste), to wytwarzanie rysunków mających na celu ułatwienie komunikacji z komputerem, a więc tworzenie menu graficznych, siatek i symboli do wykonywania rysunków lub komend, tworzenie programowych waluatorów, ikon i kursorów graficznych. Wszystkie te działania możemy określić wspólnym mianem *prac konwersacyjnych*.

Szeroki zakres wymagań, jakie te prace stwarzają, rzadko daje się spełnić korzystając z możliwości jednego urządzenia wyjściowego, a nawet jednej technologii. Dla przykładu, rysunki techniczne powinny być wykonywane na papierze lub plastyku, mieć duży rozmiar i dużą dokładność oraz wiele grubości linii (chociaż mogą być rysowane w jednym kolorze), podczas gdy przezrocza lub filmy animowane często powinny być wytworzone na negatywie o formacie 35 mm lub mniejszym i w wielu kolorach - czasem nawet z realistycznym cieniowaniem i teksturą. W żadnym z tych działań nie jest możliwe otrzymanie wyniku (produktu) w czasie rzeczywistym. Przy pracy konwersacyjnej budujemy na ekranie rysunek techniczny, modyfikujemy lub usuwamy jego części w miarę postępu pracy, a w przypadku przeglądania sekwencji animowanych oczekujemy możliwie natychmiastowej odpowiedzi. Ponadto, chcemy aby wynik na urządzeniu wyjściowym był o postaci możliwie bliskiej wynikowi przewidywanemu.

4.2. Urządzenia wyjściowe

Różnorodność prac rysunkowych i konwersacyjnych oraz ograniczenia aktualnego poziomu techniki spowodowały powstanie wielu urządzeń wyjściowych do zabezpiecze-

nia potrzeb grafiki komputerowej. Można podzielić je na dwie główne grupy

Monitory graficzne do prac konwersacyjnych w czasie rzeczywistym

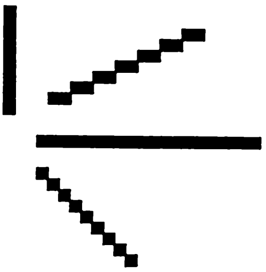
- lampy obrazowe,
- lampy obrazowe z pamięcią,
- monitory z wyświetlaniem wektorowym,
- monitory z wyświetlaniem rastrowym,
- wskaźniki plazmowe;

Urządzenia kopiujące, do produkcji kopii rysunków na papierze lub innym nośniku

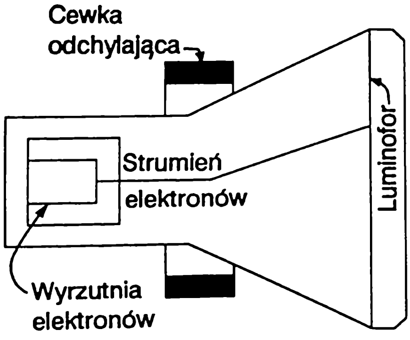
- kopiarki z ekranu,
- plotery elektrostatyczne,
- drukarki graficzne,
- plotery wektorowe,
- kamery mikrofilmowe.

Mimo znacznych różnic w konstrukcji, metodach rysowania i wyglądzie rysunku, wszystkie urządzenia mają wspólną cechę - mogą one umieszczać jedynie ograniczoną liczbę punktów na *powierzchni roboczej*. Powierzchnię roboczą można traktować jako rodzaj arkusza pokratkowanego papieru, na którym urządzenie może rysować z pewnymi ograniczeniami. Niektóre urządzenia (szczególnie plotery i lampy obrazowe z pamięcią) wykazują ograniczenia jedynie co do punktów końcowych rysunku, np. linia prosta musi zaczynać się i kończyć w kwadracie siatki. Inne urządzenia mają ograniczenia co do każdego punktu rysunku, np. każdy punkt linii musi leżeć w kwadracie siatki. Oznacza to, że linie nachylone mają "schodkową" strukturę i jedynie linie poziome lub pionowe wydają się ciągłe (rys. 4.1).

Urządzenia różnią się tzw. *rozdzielczością*, czyli gęstością siatki, na której są wykonywane rysunki. Niektóre z komputerów osobistych mają rozdzielczość jedynie 40 × 40 kwadratów. Większość pozostałych komputerów ma rozdzielczość 250 × 190, ale wcale nie rzadkie są przypadki komputerów o rozdzielczości 600 × 400. Taką roz-



Rys. 4.1. Linie złożone z pikseli



Rys. 4.2. Lampa obrazowa (CRT)

dzielczość zwykle określa się mianem "dużej rozdzielczości". Systemy profesjonalne mają rozdzielczość od 512×512 do 1280×1024 dla urządzeń rastrowych i do 4096×3120 dla urządzeń z lampą pamięciową. Wiele ploterów pisakowych charakteryzuje się rozdzielczością do około 46000×25000, natomiast niektóre plotery laserowe aż do 11×8 milionów punktów! Porównajmy te liczby z rozdzielczością domowego telewizora (około 800×500 punktów) i klatki filmu 35 mm (około 8000×5000 punktów).

Pod względem rozdzielczości urządzenia wyjściowe mogą być zaliczane do różnych klas:

zgrubnej: o rozdzielczości do około 256×256 punktów,
średniej: o rozdzielczości od około 256×256 do około 512×512 punktów,
dużej: o rozdzielczości 1024×1024 lub więcej punktów.

Na ogół, wskaźniki plazmowe i nieliczne monitory wizyjne mają rodzielczość zgrubną; większość monitorów wizyjnych ma rozdzielczość średnią; plotery, kamery mikrofilmowe, monitory do wyświetlania wektorowego, monitory z lampą pamięciową i nieliczne monitory rastrowe mają rozdzielczość dużą.

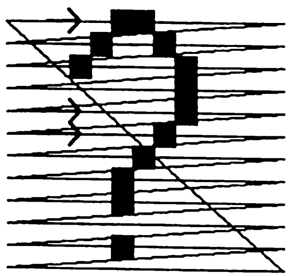
4.2.1. Monitory

Urządzenia z lampą obrazową

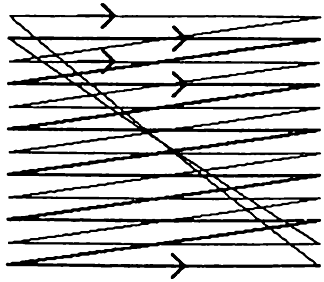
Monitory z rastrowym i wektorowym odnawianiem obrazu oraz z lampami z pamięcią używają lamp obrazowych (ang. *cathode ray tube* - CRT). *Lampa obrazowa* składa się z próżniowej bańki o płaskim dnie, tworzącym ekran (rys. 4.2). Warstwa luminoforu, nałożona na wewnętrzną powierzchnię ekranu, ma tę właściwość, że pod wpływem padającego na nią strumienia elektronów emituje światło. Luminofory mogą świecić w białym, żółtym, niebieskim, pomarańczowym i czerwonym kolorze; do produkcji lamp kolorowych używa się zestawów wielu luminoforów. Na końcu lampy, na wprost ekranu, jest umieszczona wyrzutnia elektronowa. Wytwarza ona cienki strumień elektronów, który jest sterowany i pada w określonych miejscach ekranu, powodując jego świecenie. Świecenie luminoforu nie zanika natychmiast po zaniku padającego nań strumienia elektronów, ale trwa jeszcze przez czas zależny od typu użytego luminoforu (od około 1 μs do około 0.1 s). Czas ten, na ogół nie wystarcza do zaobserwowania obrazu, ale ponieważ strumień elektronów nie ma inercji, to może być łatwo włączany i wyłączany. Poruszając się po ekranie z dużą szybkością może oświetlać ten sam punkt wiele razy na sekundę. Tak więc dobierając odpowiednio częstotliwość przebiegu strumienia po ekranie i czas poświaty luminoforu oraz biorąc pod uwagę inercję wzroku, można uzyskać niemigocący obraz, który zanika wówczas, gdy strumień elektronów zostanie wyłączony. Proces otrzymywania obrazu za pomocą ciągłego oświetlania strumieniem elektronów nazywa się *odnawianiem.*

Ruch strumienia jest sterowany elektromagnetycznie za pomocą urządzenia odchylającego. Sposób przesuwania strumienia ma zasadnicze znaczenie w tworzeniu obrazu. Powszechnie są stosowane dwa rodzaje ruchów strumienia elektronów, prowadzące do dwóch rodzin urządzeń obrazowych.

Pierwszy rodzaj to *odchylanie rastrowe*, w którym strumień przebiega ustalony wzór na ekranie, poruszając się od brzegu do brzegu i w dół ekranu, wzdłuż ustalonego przebiegu linii, powtarzanego od 50 do 60 razy na sekundę. Strumień jest włączany tylko tam, gdzie pojedynczy punkt ma być rozjaśniony (rys. 4.3). W podobny sposób pracuje znany powszechnie telewizor. W tym przypadku stumień jest rzadko całkowicie wyłączany; raczej jego intensywność jest zmieniana (modulowana) zgodnie z poziomem jasności odtwarzanego obrazu. W efekcie, linie rastra są często widoczne na ekranie czarno-białego telewizora. W celu zmniejszenia widoczności tych linii i zmniejszenia szerokości pasma sygnału wizyjnego, obrazy telewizyjne są złożone z dwóch ramek: jedna zawiera nieparzyste linie rastra, druga parzyste. Połączone ramki tworzą kompletny obraz telewizyjny, odnawiany 25 lub 30 razy na sekundę (rys. 4.4). Jedynie w systemach graficznych korzystających z odbiorników telewizyjnych używa się tej techniki zwanej *odchylaniem z przeplataniem*; większość systemów graficznych tworzy pojedynczą ramkę, zawierającą cały obraz.



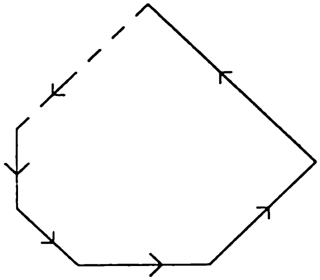
Rys. 4.3. Wyświetlanie rastrowe bez przeplatania



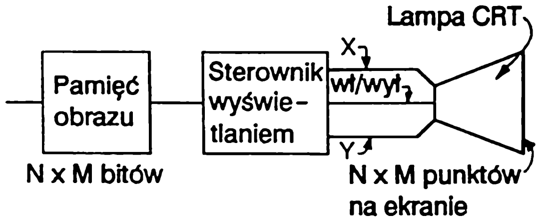
Rys. 4.4. Wyświetlanie rastrowe z przeplataniem

W drugiej metodzie odnawiania obrazu strumień porusza się bezpośrednio do punktów wymagających rozjaśnienia. Jest on wyłączany w czasie ruchu i włączany dopiero po osiągnięciu punktu docelowego (rys. 4.5). Powstający wzór jest odnawiany 50 do 60 razy na sekundę; kolejność przebieganych punktów jest określona przez kształt wyświetlanego obrazu. Ten rodzaj pracy przypomina trochę pracę oscyloskopu i znany jest pod nazwą *odchylania wektorowego* lub *przypadkowego* (co może prowadzić do nieporozumień). Termin "przypadkowy" został zapożyczony z teorii prawdopodobieństwa, gdyż twórcy systemu doszli do wniosku, że rozjaśniony jest jakiś punkt, równie dobrze jak inny dowolny.

Do utworzenia obrazu jedną z tych metod, należy przeznaczyć część systemu komputerowego do stałego odnawiania obrazu. W urządzeniach rastrowych jest to sterownik rastra, w przypadku wyświetlania wektorowego rolę tę musi pełnić sam procesor, albo specjalny układ - sterowanie wyświetlaniem (ang. *display controller*). W obydwu przypadkach tworzony obraz musi być przechowywany w układach pamięci obrazu. Tak więc w przypadku prostego systemu odnawiania obrazu mamy konfigurację podobną jak na rys. 4.6, złożoną z pamięci obrazu, sterownika wyświetlenia i lampy obrazowej.

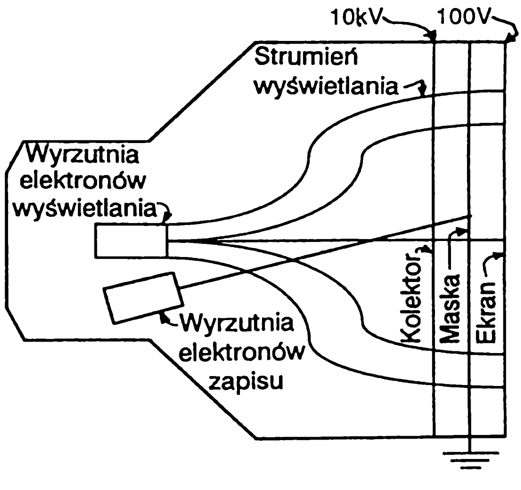


Rys. 4.5. Linia przy wyświetlaniu wektorowym



Rys. 4.6. Prosty system odnawiania obrazu monochromatycznego

Pierwsze systemy z lampą obrazową, stosowane w grafice komputerowej, pracowały w trybie wektorowego odnawiania obrazu i były drogie nie tylko z powodu rozmiarów stosowanych lamp, ale także z powodu dużych kosztów układów sterujących wyświetlaniem. Pod koniec lat sześćdziesiątych amerykańska firma Tektronix (producent oscyloskopów) wyprodukowała nową lampę obrazową, która zrewolucjonizowała dotychczasowe rozwiązania. Lampa ta nosi nazwę *lampy obrazowej z pamięcią* (ang. *direct view storage tube* - DVST). Lampa nie wymaga ciągłego odnawiania obrazu, gdyż sama jest używana jako pamięć. Obraz raz zapisany, pozostaje na ekranie, bez potrzeby dalszych akcji ze strony komputera lub jednostki sterującej. Efekt ten osiąga się dzięki temu, że strumień elektronów zapisuje obraz nie bezpośrednio na ekranie lampy, ale na siatce specjalnie wykonanych drutów, znajdującej się tuż za luminoforem. Strumień padając na siatkę tworzy wzór ładunków elektrycznych, od-



Rys. 4.7. Lampa z pamięcią (DVST)

powiadający obrazowi. Wzór ten jest przenoszony na ekran za pomocą innego strumienia elektronów, oświetlającego całą siatkę (rys. 4.7). Na początku lat siedemdziesiątych lampa ta zmniejszyła znacznie koszt urządzeń graficznych (czasem nawet dziesięciokrotne) i stała się podstawowym elementem urządzeń, których koszt jest mały i średni. Około osiem lat później rozpowszechniła się technika rastrowa.

Monitory z lampą pamięciową

Urządzenia wyświetlające z lampą pamięciową (DVST) nie potrzebują układów do przechowywania obrazu i charakteryzują się dużą rozdzielczością od 1024×768 punktów do 4096×3120 punktów - zależnie od rozmiaru ekranu. Niestety, urządzenia z lampą pamięciową pozwalają na wyświetlanie linii tylko w jednym kolorze i tylko z jednym poziomem jasności. Nie można zatem, uzyskać cieniowania z różnymi poziomami szarości lub w różnych kolorach, koniecznego w zastosowaniach obrazowych. Z tego powodu urządzenia z lampą pamięciową mają głównie zastosowania techniczne i naukowe, np. do wykonywania rysunków kreskowych (wektorowych).

Chociaż lampy z pamięcią nadają się do kreślenia linii, to jednak ich główną wadą jest niemożność selektywnego wymazywania ekranu. Za każdym razem wówczas, gdy trzeba zmodyfikować część rysunku, należy usunąć cały rysunek i narysować poprawioną wersję od nowa. Proces ponownego rysowania może zająć kilka sekund, zwłaszcza jeśli rysunek jest skomplikowany. Z tego powodu lampy obrazowe z pamięcią nie nadają się do dynamicznej zmiany obrazu. Firma Tektronix zaproponowała rozwiązanie tego problemu. W nowych lampach - zachowując właściwości pamięciowe - dodano funkcję ciągłego odnawiania obrazu; pozwala to na narysowanie do 1000 krótkich wektorów odnawialnych. Zwiększył się jednak koszt lampy. Urządzenia z lampami obrazowymi z pamięcią nie mogą współpracować z piórem świetlnym.

Monitory z wyświetlaniem wektorowym

Urządzenia z wyświetlaniem wektorowym mają wiele zalet, są jednak droższe niż urządzenia z lampą pamięciową. Mogą być używane do wykonywania rysunków o wysokiej jakości, z liniami o różnych grubościach i kolorach; można wyświetlać obszary cieniowane; przede wszystkim jednak pozwalają na dynamiczne tworzenie, częściowe wymazywanie i odtwarzanie obrazów. Połączone z szybkim procesorem graficznym umożliwiają przesuwanie, w czasie rzeczywistym, obiektów graficznych po ekranie, zmianę ich skali lub orientacji - co pozwala na animację obrazów. Ponieważ cały obraz jest odnawiany 50 lub 60 razy na sekundę, to każda jego część może być selektywnie wymazana lub zmieniona podczas procesu odnawiania. Operację wymazania lub zmiany wykonuje program samodzielnie lub na żądanie użytkownika, wprowadzone za pomocą pióra świetlnego lub innego podobnego urządzenia. Podobnie jak lampy pamięciowe, urządzenia z wektorowym wyświetlaniem obrazu mają dużą rozdzielczość. W przeciwieństwie do lamp pamięciowych, mają tę wadę, że w miarę przybywania linii na ekranie, zbliżając się do dopuszczalnego limitu, procesor graficzny ma coraz więcej pracy i nie może nadążyć z odnawianiem obrazu. Pojawia się denerwujące *migotanie*. Jeszcze kilka lat temu migotanie występowało, gdy liczba wyświetlanych wektorów wcale nie była duża, co nie przysporzyło sławy tym urządzeniom. Obecnie są dostępne

niezbyt drogie urządzenia wyświetlające do 40 000 wektorów bez migotania. Można by sądzić, że taka liczba wektorów wystarczy całkowicie do poradzenia sobie z każdym, nawet najbardziej złożonym rysunkiem. Trzeba jednak pamiętać, że okręgi i inne krzywe muszą byc składane z bardzo krótkich wektorów tak, aby ich linie były gładkie. Zwykle linię okręgu składa się ze 150 lub nawet 250 wektorów; można zatem wyświetlić około 200 okręgów bez migotania.

Monitory z wyświetlaniem rastrowym

Do większości współczesnych prac sprzętowych i programowych w dziedzinie grafiki komputerowej używa się urządzeń z wyświetlaniem rastrowym. W urządzeniach tych strumień elektronów przesuwa się po ekranie wzdłuż ustalonego wzoru; przechodzi przez każdy punkt i rozjaśnia te punkty, które tworzą obraz. System rastrowy jest tani, zwłaszcza jeśli korzysta ze standardowego telewizora lub monitora telewizyjnego. Tanie urządzenia z wyświetlaniem rastrowym wykorzystują standard telewizyjny; mają tylko 625 linii w standardzie europejskim (525 linii w standardzie amerykańskim), a zatem mają małą rozdzielczość. Co więcej, ze względów techicznych nie wszystkie linie mogą być wyświetlone, więc faktycznie używa się jedynie 90% dostępnych linii. Urządzenia w standardzie amerykańskim mogą wyświetlić około 480 × 600 punktów, a w europejskim około 575 × 760 punktów.

Do zastosowań w grafice komputerowej i w telewizji produkuje się drogie lampy o dużej rozdzielczości dające ponad 1000 linii w pionie i ponad 1000 punktów w poziomie. Tańsze urządzenia do wyświetlania rastrowego mają rozdzielczość 256 × 256 lub 512 × 512 punktów. Systemy rastrowe są niezawodne; łatwo daje się narysować kolorowe linie lub obszary; możliwa jest animacja oraz - w przeciwieństwie do urządzeń z wyświetlaniem wektorowym - nie ma migotania obrazu, niezależnie od jego zawartości informacyjnej. Jednakże wadą systemu rastrowego o małej rozdzielczości jest, wspominany już, efekt "schodkowy" linii, który można usunąć metodami programowymi lub sprzętowymi.

Punkty wyświetlane na ekranie są to małe kwadraty lub prostokąty, zwykle tak ustawione, że każdemu z nich odpowiada jedno miejsce w pamięci: taką organizację pamięci określa się mianem *odwzorowania bitowego*. Ponieważ każde miejsce w pamięci może zawierać tylko zero (piksel nie rozjaśniony) lub jedynkę (piksel rozjaśniony), to 512 × 512 komórek pamięci (32 KB) wystarcza do zapamiętania jednokolorowego obrazu 512 × 512 punktów. Dla obrazów wielokolorowych są potrzebne pamięci o większej pojemności. Niektóre systemy zawierają aż do 512 × 512 × 24 komórek pamięci, co odpowiada pamięci o pojemności 0.75 MB przeznaczonej na sam obraz. Pamięć ta jest budowana z układów scalonych pamięci półprzewodnikowych i nosi nazwę *bufora ramki* (lub bufora obrazu). Zadaniem bufora jest przyjmowanie obrazu od procesora i przedstawianie go na ekranie. W realizacji tego zadania pomaga sterownik wyświetlaniem.

Widzenie kolorów

Oko ludzkie jest czułe na światło o długości fali od okolo 380 nm do okolo 780 nm (1nm $= 10^{-9}$ m). Fale o mniejszej długości są spostrzegane jako fiolet, większej jako

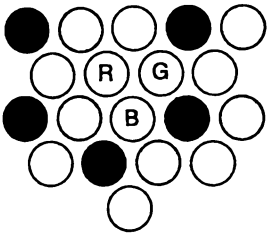
czerwień. Przy normalnych warunkach oświetleniowych oko ludzkie jest najbardziej czułe na światło o długości fali 555 nm, widziane jako barwa zielono-żółta.

W odpowiedzi na światło, oko ludzkie działa jak mała sferyczna kamera. Z jednej strony ma ono soczewkę i przysłonę (tęczówka), z drugiej elementy światłoczułe (siatkówka). Siatkówka składa się z dwóch typów elementów światłoczułych: *pręcików* wrażliwych przy małych poziomach jasności oraz *czopków* działających przy dużych poziomach jasności i służących do rozpoznawania kolorów. Elementy te są połączone z nerwem wzrokowym, przekazującym informacje do mózgu w celu zinterpretowania. W oku znajduje się około 130 milionów pręcików i czopków. Nie są one rozmieszczone równomiernie na siatkówce; ponadto pręcików jest około 17 razy więcej niż czopków. Większość czopków znajduje się w jednym miejscu zwanym *żółtą plamką*, na której w ogóle nie ma pręcików. Warunkiem wyraźnego widzenia jakiegokolwiek obiektu jest ustawienie jego obrazu na żółtej plamce. Człowiek wykonuje ten proces nieświadomie, lekko poruszając okiem.

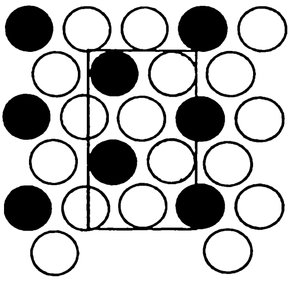
W oku występują trzy typy czopków czułe na światło: niebieskie, zielone i czerwone. Połączone sygnały z tych czopków pozwalają na poprawne odbieranie kolorów. Wszystkie widziane przez nas kolory są kombinacją kolorów: niebieskiego, zielonego i czerwonego.

Kolor w telewizji i monitorach rastrowych

Podobnie jak oko rozpoznaje kolory przez współdziałanie elementów czułych na barwy: niebieską, zieloną i czerwoną, tak i telewizor kolorowy tworzy obraz barwny przez selektywne rozjaśnianie luminoforów: niebieskiego, zielonego i czerwonego. Luminofory są rozmieszczone na ekranie w zestawach po trzy kolorowe kropki, "pastylki" ustawione w wierzchołkach trójkąta (rys. 4.8). Ponieważ kropki są małe i blisko siebie ustawione, to oko - z normalnej odległości patrzenia - nie jest w stanie ich zobaczyć jako oddzielne punkty. Tak więc gdy sąsiednie kropki są rozjaśnione, widzimy je jako jedną plamkę o wypadkowym kolorze. Jeden piksel może składać się z jednej takiej *triady luminoforów* lub z ich grupy (rys. 4.9). Urządzenia o mniejszej rozdzielczości mają większe grupy triad.



Rys. 4.8. Trójkątny wzór kropek luminoforu; B - niebieski, G - zielony, R - czerwony



Rys. 4.9. Grupa kropek luminoforu tworzących piksel; (ustawienie kropek tworzy piksel, który nie jest kwadratem)

Trzy wyrzutnie elektronów, po jednej dla każdego koloru, są dokładnie zogniskowane na odpowiednich kolorowych kropkach i rozjaśniają je z różną intensywnością. W odbiorniku telewizyjnym intensywność jest zmieniana w sposób ciągły przez układy analogowe, natomiast w urządzeniach grafiki komputerowej sterowanie jest cyfrowe i odbywa sie skokowo. W prostych systemach wyświetlania, gradacja tych skoków jest zgrubna, czasem ograniczona tylko do włączania i wyłączania strumienia. Trzy strumienie pozwalają uzyskać tylko osiem możliwych kombinacji "włączony-wyłączony", zatem można zamodelować tylko osiem kolorów w tym biel i czerń. Osiągalne kolory przedstawiono w tabl. 4.1.

Tablica 4.1.

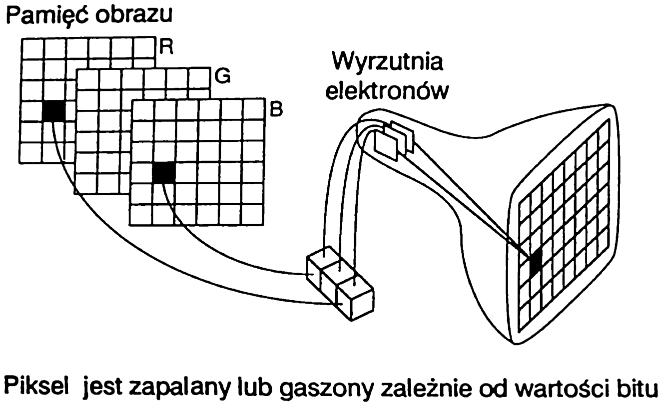
Kolor	Strumień czerwony	Strumień zielony	Strumień niebieski
Czarny	wyłączony	wyłączony	wyłączony
Niebieski	wyłączony	wyłączony	włączony
Zielony	wyłączony	włączony	wyłączony
Zielono-niebieski	wyłączony	włączony	włączony
Czerwony	włączony	wyłączony	wyłączony
Purpurowy	włączony	wyłączony	włączony
Żółty	włączony	włączony	wyłączony
Biały	włączony	włączony	włączony

Do manipulacji kolorami i ich zapamiętania należy przeznaczyć część komputera w celu przechowywania reprezentacji pikseli. Do przechowywania informacji o obrazie (monochromatycznym) 512×512 pikseli potrzebna jest pamięć o pojemności 32 KB. Do realizacji obrazu monochromatycznego jest potrzebny jeden bit na piksel. Natomiast do uzyskania do obrazu ośmiokolorowego są potrzebne trzy bity na piksel, które są często ustawione w oddzielnych dla każdego koloru *płaszczyznach bitowych* (ang. *bit-plane*) (rys. 4.10), a ich wartość określa wyświetlany kolor. Przy założeniu, że dla bitu o wartości 1 strumień jest włączony, dla 0 wyłączony, zapis w tabl. 4.1 ulegnie przekształceniu i będzie taki jak w tabl. 4.2.

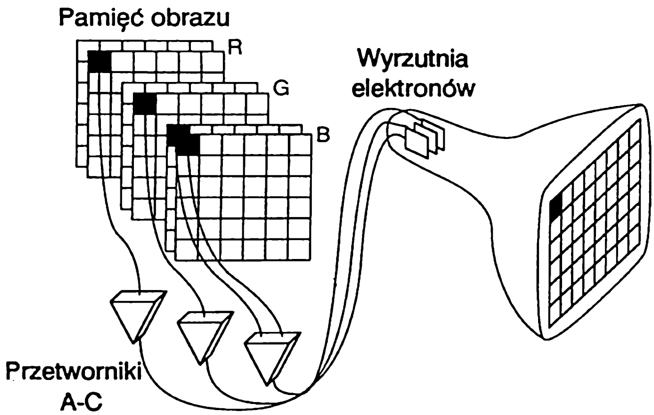
Tablica 4.2.

Kolor	Płaszczyzna bitowa 1	Płaszczyzna bitowa 2	Płaszczyzna bitowa 3
Czarny	0	0	0
Niebieski	0	0	1
Zielony	0	1	0
Zielono-niebieski	0	1	1
Czerwony	1	0	0
Purpurowy	1	0	1
Żółty	1	1	0
Biały	1	1	1

Do uzyskania, więcej niż ośmiu kolorów, należy zastosować więcej płaszczyzn bitowych. Jeżeli każdemu strumieniowi zostaną przypisane dwa bity do kontroli jego



Rys. 4.10. System graficzny z 8 kolorami (3 płaszczyzny bitowe)



Rys. 4.11. System graficzny z 64 kolorami (6 płaszczyzn bitowych)

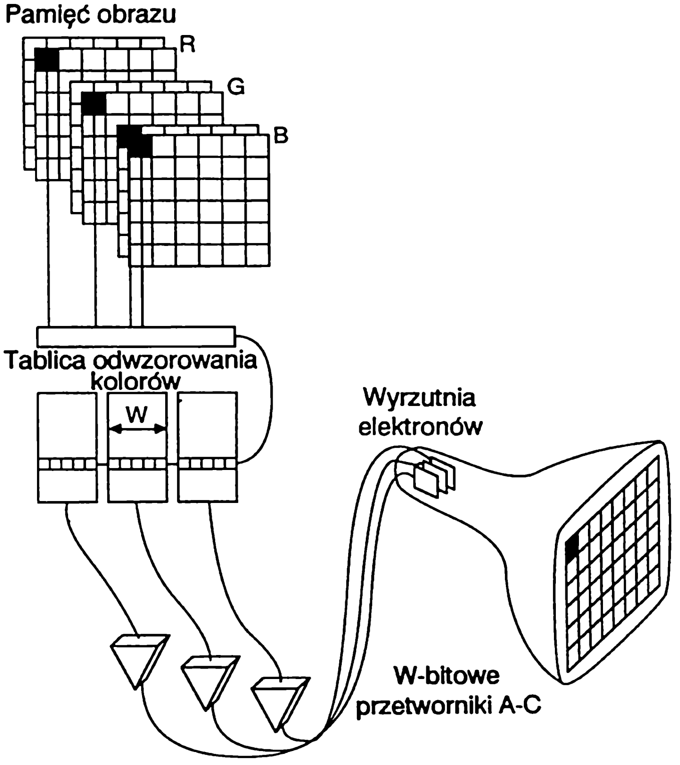
jasności za pośrednictwem przetwornika analogowo-cyfrowego, to można wyświetlić 64 kolory (rys. 4.11). Liczba kolorów wynika z kombinacji możliwych wartości bitów. Do uzyskania większej liczby kolorów należy użyć większej liczby płaszczyzn, chociaż na ogół nie zdarzają się systemy o większej liczbie płaszczyzn niż 24. Konfiguracja pamięci obrazu z 24 płaszczyznami bitowymi jest droga, ale pozwala na uzyskanie 16 777 216 różnych kolorów. W przypadku gdyby nawet każdy piksel ekranu o rozmiarach 1024×1024 był rozjaśniony na inny kolor, to i tak nie można uzyskać jednocześnie ponad 16 milionów różnych kolorów. W praktyce, w jednym obrazie stosuje się ograniczenie do znacznie mniejszego zbioru kolorów z takiej palety. Technika pozwa-

lająca na zmniejszenie pamięci, a jednocześnie na uzyskanie szerokiego zakresu kolorów (dużej palety kolorów), nosi nazwę tablicy odwzorowania kolorów.

Tablica odwzorowania kolorów

Przy użyciu trzech płaszczyzn pamięciowych można uzykać na ekranie tylko osiem kolorów. Jeśli jednak sprzęt zostanie uzupełniony ośmioelementową 24-bitową tablicą, której bity będą używane do sterowania strumieni elektronów, to wartości w płaszczyznach bitowych będą wskazywały elementy tej tablicy. Za pomocą tych elementów można wybierać dowolne osiem spośród dostępnych 16 milionów kolorów (rys. 4.12). Komórki płaszczyzn bitowych nadal będą odwzorowywać piksele na ekranie.

W wielu systemach profesjonalnych stosuje się 12 płaszczyzn pamięciowych i 24-bitowe tablice odwzorowania kolorów. Umożliwia to wybór - w każdym obrazie - 4096 kolorów z pełnej palety 16 milionów kolorów. W takiej konfiguracji do zapamiętania obrazu 512×512 pikseli potrzeba 384 KB pamięci na sam obraz i 12 KB na tablicę kolorów. W niektórych systemach osobistych stosuje się mniejszą paletę i tab-



Rys. 4.12. Zmiana kolorów za pomocą tablicy odwzorowania

licę odwzorowania. Popularna konfiguracja pozwala na uzyskanie 256 kolorów z palety 4096 kolorów, przy zastosowaniu 12-bitowej tablicy kolorów i 8-bitowej pamięci obrazu. W tym przypadku obraz 512 × 512 pikseli wymaga pamięci o pojemności 256 KB na zapamiętanie obrazu i pamięci o pojemności 384 bajty na tablicę kolorów.

Zależności wiążące liczbę kolorów i płaszczyzn pamięci:
- liczba kolorów w palecie = 2^W
- liczba wyświetlanych kolorów = 2^N

gdzie: W - liczba bitów tablicy odwzorowania kolorów, N - liczba płaszczyzn bitowych.

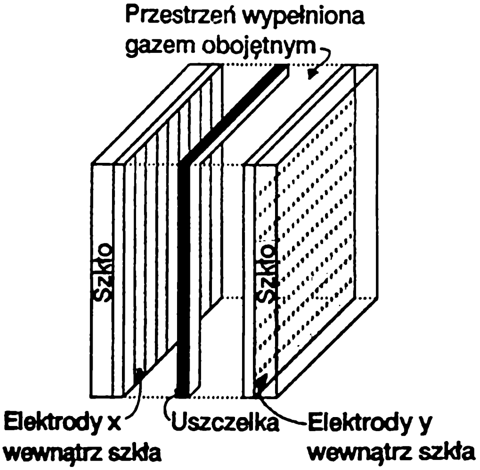
Tablice odwzorowania kolorów nie tylko zmniejszają rozmiar potrzebnej pamięci, ale także mogą być używane do uzyskiwania ograniczonej formy animacji. Osiąga się to, przez dynamiczną zmianę kolorów w tablicy. Zmiana koloru tła niektórych elementów rysunku na kolor im właściwy, wywołuje ich pozorny ruch na ekranie.

Wskaźniki plazmowe

Wskaźniki plazmowe są zbudowane na całkowicie innej zasadzie niż lampy obrazowe. Nie wykorzystują one właściwości luminoforów, a ich działanie wykorzystuje zjawisko świecenia obojętnego gazu (np. neonu) w polu elektrycznym. Istnieją dwa typy wskaźników plazmowych:

- wskaźniki stałoprądowe, przeznaczone dla mniejszych monitorów alfanumerycznych
- wskaźniki zmiennoprądowe, używane do obrazów tekstowych i graficznych.

Wskaźniki zmiennoprądowe zawierają cienką warstwę neonu, umieszczoną między dwoma płytami szklanymi. Na jednej z płyt są naniesione cienkie poziome paski elektrod, a na drugiej pionowe (rys. 4.13). Przepływ prądu zmiennego przez elektrody x i y powoduje świecenie gazu na skrzyżowaniach pasków. Tak uaktywniony piksel będzie się świecił bez odnawiania, tak długo, aż podejmie się akcję zgaszenia go.



Rys. 4.13. Zmiennoprądowy wskaźnik plazmowy z przezroczystymi elektrodami wzdłuż osi X i Y

Wskaźniki zmiennoprądowe w postaci cienkich paneli są produkowane w rozmiarach i o rozdzielczości, które sprawiają, że są one praktyczną alternatywą dla lamp obrazowych. Są już dostępne wskaźniki mające 768×960 monochromatycznych pikseli, każdy o średnicy około 0.3 mm, ale ich użycie w grafice nie jest zbyt powszechne. Wydaje się, że zarówno wskaźniki plazmowe jak i inne płaskie wskaźniki będą miały znaczny wpływ na systemy graficzne w przyszłości i mogą zagrozić obecnej supremacji monitorów ekranowych.

4.2.2. Graficzne urządzenia kopiujące

Niezależnie od możliwości obejrzenia rysunku na ekranie monitora, jest konieczne otrzymanie jego kopii na papierze. Tak więc wraz z rozwojem różnych typów wskaźników graficznych, rozwijały się także urządzenia do otrzymywania kopii, szczególnie kolorowych rysunków. Graficzne urządzenia kopiujące dzielą się na pięć głównych kategorii:

- kopiarki z ekranu,
- plotery elektrostatyczne,
- drukarki graficzne,
- plotery wektorowe,
- urządzenia do zapisu na mikrofilmie.

Każde z tych urządzeń wykorzystuje specyficzną technologię i ma pewien zakres rozdzielczości. Ponieważ w większości są to urządzenia elektromechaniczne, zwykle są one droższe niż urządzenia wyświetlające o zbliżonej klasie.

Kopiarki z ekranu

Trudno jest wykonać dobre, kolorowe zdjęcie bezpośrednio z ekranu rastrowego wskaźnika graficznego. Najczęściej powierzchnia ekranu wyświetlającego jest wypukła, co powoduje zniekształcenie zdjęcia. Krzywizna jest ledwo zauważana przy oglądaniu ekranu, ale wydaje się, że jest przejaskrawiona na zdjęciu. Czułość filmów kolorowych jest dobrana do średniego oświetlenia dziennego lub do światła sztucznego, co powoduje złe naświetlenie pewnych kolorów z ekranu. Ponadto ustawienie aparatu, zwłaszcza przy długich czasach naświetlania, utrudnia normalne korzystanie z monitora.
W ciągu kilku ostatnich lat pojawiły się specjalne aparaty fotograficzne do kopiowania z ekranu, które pozwoliły wreszcie na rozwiązanie wyżej wymienionych problemów. Oryginalne prace badawcze w tej dziedzinie prowadziły firmy Polaroid Corporation i Dunn Instruments Inc. i z tego powodu takie aparaty często noszą nazwę "Dunn Cameras". Urządzenia do kopiowania z ekranu mają różny wygląd zewnętrzny, ale wszystkie mają zbliżoną konstrukcję i zasadę działania. Szczelne pudełko, zawiera małą monochromatyczną lampę obrazową, z trzema kolorowymi filtrami z przodu i z aparatem fotograficznym zamocowanym przed tym zestawem; całość jest sterowana za pomocą mikroprocesora. Przy wykonywaniu zdjęcia w urządzeniu kopiującym jest używany ten sam sygnał wizyjny, który jest wysyłany na ekran monitora głównego.

W urządzeniu kopiującym najpierw jest pobierany sygnał czerwony i fotografowany przez filtr czerwony, potem sygnał zielony - fotografowany przez filtr zielony, a w końcu sygnał niebieski - fotografowany przez filtr niebieski. Wszystkie te trzy elementy zdjęcia są nakładane na jedną klatkę filmu, co pozwala na uzyskanie w pełni kolorowego zdjęcia. Cały ten proces trwa około jednej minuty, zależnie od rodzaju użytego filmu. Fakt, że każda składowa obrazu jest naświetlana oddzielnie pozwala na dobranie czasu naświetlania do czułości filmu dla każdej barwy. Czas naświetlania dla barwy czerwonej jest około 3 - 4 razy dłuższy niz dla innych barw. Ta właściwość wybiórczej kontroli czasu naświetlenia i stosowanie płaskiej lampy dają znakomicie nasycone, kolorowe zdjęcia o wysokiej jakości. Dodatkową zaletą jest możliwość lekkiej zmiany proporcji barw, bez potrzeby powtórnego tworzenia samego obrazu. Niektóre z tych urządzeń umożliwiają także wykonywanie kolorowych zdjęć z ekranu monochromatycznego.

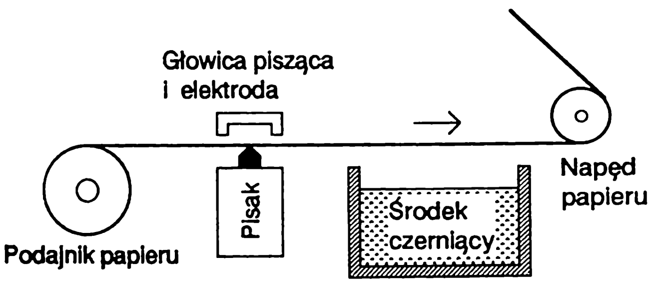
Urządzenia do kopiowania z ekranu wykorzystują różne formaty filmów:

- negatywowe Polaroid 8 × 10 cali (20 × 25 cm) i 4 × 5 cali (10 × 13 cm),
- przezrocza 8 × 10 cali (20 × 25 cm),
- przezrocza 35 mm,
- kinematograficzne 16 mm i 35 mm (zastosowanie tylko w niektórych urządzeniach kopiujących).

Negatywy dużych formatów są drogie i trudne w procesie wywoływania. Urządzenia do kopiowania z ekranu są jedynymi urządzeniami dokładnie kopiującymi zawartość ekranu. Przykładem kopiarek innego typu są przyrządy do kopiowania czarno-białego lub w pewnej skali szarości, bezpośrednio z ekranu lub z pamięci obrazu. W dawniejszych wersjach tych urządzeń otrzymywano kopię na papierze światłoczułym, który był przesuwany przed ekranem wewnętrznej lampy obrazowej. Następnie papier śwatłoczuły był utrwalany termicznie. Papiery światłoczułe są dosyć drogie, ale świetnie nadają się do szybkiego wykonywania kopii próbnych.

Plotery elektrostatyczne

Plotery te wytwarzają obraz w sposób elektrostatyczny, na podłożu pokrytym materiałem dielektrycznym. Obraz jest przenoszony na papier lub plastyk za pomocą sze-



Rys. 4.14. Ploter elektrostatyczny

regu maleńkich pisaków (rys. 4.14). Komputer używa ich do nanoszenia na podłoże małych, elektrostatycznych kropek, ustawionych w siatce rastra o gęstości od 100 do 200 punktów na cal. Tak naładowany papier przechodzi następnie przez zbiornik środka czerniącego (ang. *toner*), który przyczepia się do papieru we wszystkich miejscach naelektryzowanych, tworząc w efekcie obraz. Plotery pozwalają na uzyskiwanie czarno-białych rysunków o wysokiej jakości na papierze w różnych formatach. Ostatnio pojawiły się urządzenia do produkcji kolorowych rysunków, ale są one bardzo drogie. Obraz jest tworzony kolejno wierszami od góry do dołu i musi być przesyłany do plotera w sposób rastrowy. Przy obliczaniu całkowitego czasu niezbędnego do wyprodukowania kopii rysunku, trzeba wziąć pod uwagę niemały czas, jaki jest potrzebny komputerowi na utworzenie rastrowego wzoru rysunku. W przypadku małych komputerów może to trwać dość długo. Plotery elektrostatyczne pracują szybko, zwłaszcza w przypadku wykonywania wielu kopii tego samego rysunku, wówczas rastrowy wzór rysunku jest tworzony tylko raz. Przykładowo rysunek o rozmiarze (914 × 1143 mm), zawierający około 15 000 linii jest wykonywany w ciągu około 45 s wliczając w to tworzenie rastrowego wzoru rysunku przez komputer. Wykonanie takiego rysunku na szybkim ploterze pisakowym zajmuje ponad 10 minut. Ceny ploterów elektrostatycznych są wysokie i nie pojawią się one prędko w zastosowaniach komputerów osobistych.

Drukarki graficzne

Drukarki graficzne tworzą obraz rastrowy uderzając szpilkami przez taśmę barwiącą, o papier, podobnie jak normalne drukarki znakowe. Ostatnio opracowane, nowe drukarki tworzą obraz o ograniczonym zestawie kolorów, uzyskany za pomocą taśmy barwiącej. Poszczególne modele różnią się jedynie liczbą uzyskiwanych barw i sposobem prowadzenia taśmy.

W niektórych modelach taśma jest podzielona w poziomie na cztery paski: żółty, purpurowy, niebiesko-zielony i czarny. Głowica drukująca przechodzi nad każdym paskiem tyle razy, ile to jest niezbędne do utworzenia rysunku z ograniczonego zestawu kolorów; nie tworzą się żadne kolory mieszane. W innych modelach drukarek graficznych stosuje się czterokolorową taśmę podzieloną na sekcje kolorowe wzdłuż taśmy. Drukarka jest połączona bezpośrednio z ekranem i tworzy rysunek kolejno kolorami: żółtym, purpurowym, niebiesko-zielonym, na końcu czarnym. Nakładanie kolorów pozwala na uzyskanie barw: czerwonej, zielonej i niebieskiej. Jeszcze innym rozwiązaniem jest stosowanie oddzielnej kasety z taśmą dla każdego koloru, co pozwala na wymianę taśm najczęściej używanych.

Drukarki graficzne są dość tanie, zwłaszcza, że mogą być także używane jako zwykłe drukarki tekstowe. Nie można ich traktować jako prawdziwe urządzenia wyjściowe wówczas, gdy zachodzi konieczność wiernego odtworzenia kolorów. Można rozszerzyć ich paletę kolorów pozornych, tworząc programowo wzory wypełniające. Dostępne drukarki graficzne nadają się do tworzenia diagramów i wykresów, w mniejszym zakresie do złożonych obrazów graficznych.

Plotery wektorowe

Plotery wektorowe dzieli się na trzy klasy:

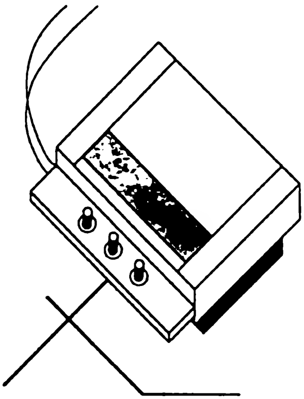
- plotery pisakowe,
- plotery ze strumieniem tuszu,
- plotery laserowe.

Plotery pisakowe są pierwszymi urządzeniami do tworzenia kopii obrazu graficznego. Powstały w wyniku rozwoju pisaków analogowych X-Y, powszechnie stosowanych w przemyśle i laboratoriach. Dostępna jest duża liczba typów tych urządzeń (różnych producentów), ale w zasadzie są istotne trzy ich wykonania:

- obrotowe lub bębnowe,
- z płaskim stołem,
- "żółwiowe".

W *ploterach bębnowych* papier lub inny nośnik jest przesuwany przez bęben w przód i wstecz pod pisakiem, poruszającym się w lewo i w prawo. Połączenie tych ruchów oraz sterowanego podnoszenia i opuszczania pisaka umożliwia wykreślenie rysunku. W *ploterach z płaskim stołem* papier jest zamocowany na płycie, pisak porusza się na ramieniu w przód i wstecz; ramię porusza się w lewo i w prawo. W ploterach "*żółwiowych*" pisak jest przenoszony na małym wózku (rys. 4.15), który - mimo, że jest dołączony do komputera za pomocą kabla - może swobodnie poruszać się po papierze, ułożonym na płaskim podłożu.

Pisak we wszystkich ploterach może poruszać się w ośmiu kierunkach, co 45° . Właściwe rysowanie jest zagwarantowane oprogramowaniem, które składa ruchy cząstkowe.



Rys. 4.15. Ploter wózkowy (żółwiowy)

Plotery można podzielić na kategorie, zależnie od ich: rozdzielczości, stopnia inteligencji, rozmiaru wykonywanego rysunku i ceny. Rozdzielczość ploterów waha się od 0.001 do 0.01 cala (0.025 do 0.25 mm), przy czym im większa rozdzielczość, tym lepszy rysunek, lecz większe obciążenie komputera i proces kreślenia wolniejszy. Jeśli jednak ploter ma wbudowany procesor (własną "inteligencję"), to zwykle jest możliwe nastawienie rozdzielczości rysowania zarówno linii jak i tekstu. Rysunki próbne można

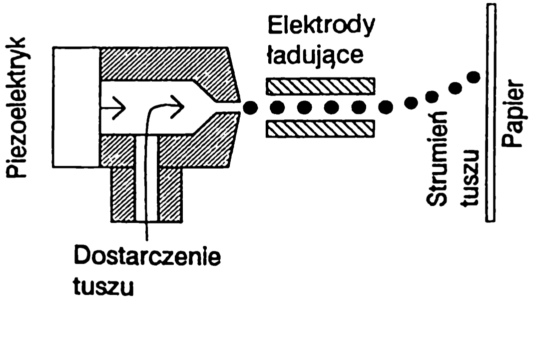
wówczas wykonać z mniejszą rozdzielczością, a końcowy produkt wykonać z pełną rozdzielczością. Ponadto, wiele ploterów inteligentnych ma zaprogramowane funkcje rysowania takich elementów, jak: okręgi, łuki itp., wywoływanych jednym poleceniem z komputera. Takie możliwości oraz duża pojemność pamięci pozwalają na przesłanie z komputera do plotera dużej części rysunku. Komputer, w czasie gdy ploter kreśli rysunek, może zająć się inną pracą. Niektóre plotery mogą być sterowane z oddzielnej jednostki, w której nośnikiem informacji jest taśma magnetyczna. Pracują wówczas niezależnie od komputera, ale rozwiązanie to zwiększa koszt zestawu.

Szerokość nośnika, na którym jest kreślony rysunek wynosi od 10 cali (250 mm) do 72 cali (1800 mm). Zarówno do ploterów bębnowych, jak i z płaskim stołem, można używać papieru w rolkach, co pozwala na kreślenie rysunków o właściwej szerokości i dowolnej długości. Jedynie do ploterów z płaskim stołem można stosować wstępnie nadrukowany papier, co jest szczególnie użyteczne przy wykonywaniu rysunków technicznych. Jako nośnika do rysowania można używać zarówno papieru jak i folii typu Mylar lub innych materiałów.

Popularne są plotery z wieloma pisakami, co pozwala na rysowanie różnych grubości linii lub w różnych kolorach. Zwykle plotery mają 2, 6 lub 8 pisaków różnego typu: z końcówką filcową (mazak), długopis lub pisak kreślarski, napełniany tuszem. Te ostatnie dają najlepszą jakość rysunku (duży wybór grubości linii), ale często zasychają i są niewygodne w użyciu.

Koszt plotera zależy od rozmiaru i funkcjonalności, ale nawet najmniejszy ploter bębnowy lub z płaskim stołem, może kosztować tyle co komputer osobisty. Plotery "żółwiowe" są tanie, nie mogą być jednak stosowane do wykonywania profesjonalnych rysunków. Ogólnie można stwierdzić, że przy tych samych rozmiarach plotery pisakowe są tańsze niż elektrostatyczne.

Plotery ze strumieniem tuszu działają z wykorzystaniem ciągłego strumienia naelektryzowanych kropelek tuszu, które są odchylane w polu elektrostatycznym (rys. 4.16). Jednostka sterująca włącza lub wyłącza pole elektrostatyczne, zależnie od tego czy tusz ma trafić na papier czy nie. Urządzenia te są szybkie i pracują cicho, ale dają ograniczoną gamę kolorów. Ponieważ są prowadzone prace badawcze w tej dzie-



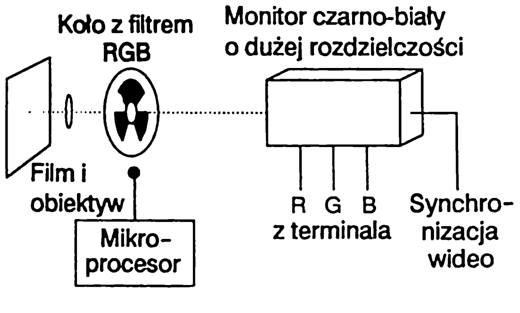
Rys. 4.16. Konstrukcja plotera ze strumienim tuszu

dzinie, należy więc oczekiwać, że takie plotery staną się niedługo konkurencyjne z urządzeniami do kopiowania z ekranu - zwłaszcza w dziedzinie zastosowań nieprofesjonalnych. Są one stosunkowo niedrogie - najtańsze kosztują tyle samo, co drukarka dobrej klasy. Główną wadą tych ploterów jest zasychanie tuszu; wymagają one stałych zabiegów konserwacyjnych.

Plotery laserowe reprodukują rysunek na powierzchnię światłoczułego bębna za pomocą promienia laserowego. Uczulone obszary przyciągają kolorowy proszek tonujący, jak w zwykłej kolorowej kopiarce elektrostatycznej. Proces ten jest powtarzany trzy razy, raz dla każdego koloru podstawowego (zielono-niebieski, purpurowy, żółty), co pozwala na uzyskanie barw pośrednich, podobnie jak w drukarkach kolorowych. Plotery laserowe są szybkie, pracują cicho i mogą być używane jako drukarki tekstowe wysokiej klasy. Są one jednak dość drogie. Nowe monochromatyczne plotery, są przeznaczone do komputerów osobistych.

Urządzenia do zapisu na mikrofilmie

Urządzenia te mają taką samą konstrukcję jak urządzenia do kopiowania z ekranu; naprzeciw ekranu monochromatycznej lampy obrazowej o dużej rozdzielczości jest ustawiona kamera. Ekran jest filmowany kolejno przez trzy kolorowe filtry (rys. 4.17). Proces filmowania jest sterowany mikroprocesorem. Urządzenia do zapisu na mikrofimie tworzą obraz bezpośrednio na podstawie instrukcji z komputera i nie zależą od urządzeń wyświetlających dołączanych do systemu użytkownika. Dają bezpośrednio na filmie o różnym formacie kolorowy obraz o dużej rozdzielczości. Dostępne są urządzenia o formacie 35 mm (film lub pojedyncze zdjęcia), 16 mm (mikrofiszki lub mikrofilmy). Każdy z tych formatów wymaga własnej kamery, która zwykle kosztuje drożej niż komputer osobisty.



Rys. 4.17. Urządzenie do zapisu na mikrofilmie

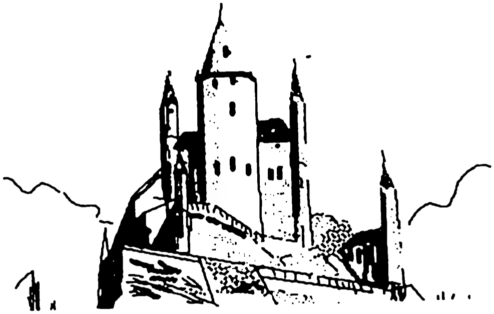
Urządzenia te są przeznaczone do poważnych prac graficznych dobrze spisują się w rastrowym i wektorowym trybie pracy, gdyż korzystają z własnego mikroprocesora. Ich obecna cena i rozmiary powodują, że są to urządzenia rzadko spotykane poza profesjonalnymi pracowniami komputerowymi.

5. Programowanie w grafice komputerowej

Znaczenie matematyki

Ku utrapieniu tych, którzy zawsze uważali matematykę za naukę tajemniczą, nie da się skutecznie programować w dziedzinie grafiki komputerowej (z wyjątkiem grafiki mozaikowej) bez znajomości co najmniej geometrii, trygonometrii i algebry. Do zrozumienia zagadnień poruszanych w książce nie jest wymagany wysoki poziom znajomości matematyki. Parę godzin pracy, dokładne przerobienie kilku następnych rozdziałów i Dodatku 1 umożliwi napisanie działających programów i studiowanie bardziej zaawansowanych podręczników.

Matematyka potrzebna jest z dwóch powodów. Po pierwsze, do zrobienia opisu szczegółów rysunku, który chcemy wykonać i przesłać do komputera. Proces ten nosi nazwę *modelowania geometrycznego*. Po drugie, do manipulowania reprezenatacją tych obiektów wewnątrz komputera i do wyświetlenia ich zgodnie z naszą wolą. Jest całkowicie możliwe opracowanie systemu graficznego, pozwalającego na tworzenie rysunków za pomocą tablicy graficznej i wskaźnika, co eliminuje proces modelowania geometrycznego. Takie systemy noszą nazwę *systemów malarskich*; wykorzystują one program, który przekształca ruchy wskaźnika (myszy lub rysika) w linie na ekranie.

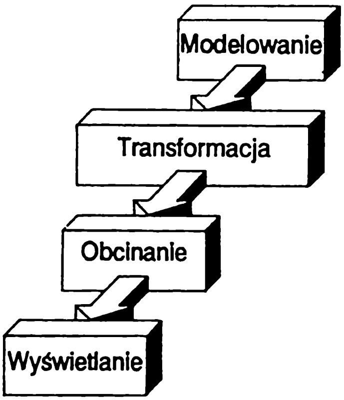


Rys. 5.1. Obraz "malowany"

Obiekt można przedstawić w postaci szkicu. System umożliwia rysowanie linii prostych, okręgów, elips oraz zamalowywanie obszarów lub usuwanie części rysunku. Tak wykonana grafika (rys. 5.1) może być całkiem efektowna i mieć dużą wartość artystyczną, do jej tworzenia są wymagane zdolności artystyczne i manualne. Dalsze przetwarzanie obrazów utworzonych za pomocą programów malarskich jest ograniczone do takich samych czynności, jakie wykonuje się przy obróbce fotografii. Oznacza to, że system malarski może powiększyć lub zmniejszyć obraz, obrócić go lub zmienić kolory. Nie może jednak zmienić żadnej z jego zasadniczych cech. Jeżeli na przykład został wykonany boczny widok domu, to nie można zażądać od systemu malarskiego pokazania widoku z przodu lub w perspektywie. System nie zna pojęć związanych z trójwymiarową naturą obiektów przedstawionego szkicu. Innymi słowy, system nie ma modelu wyświetlanego obiektu.

W celu uzyskania dowolnego widoku przetwarzanego obiektu należy wykonać specjalne zadania, z których każde wymaga użycia matematyki (rys. 5.2). Są to:

- modelowanie - polegające na dostatecznie szczegółowym opisaniu obiektu, aby komputer był w stanie "zrozumieć" jego geometrię; ten proces jest nazywany opisem obiektu;
- transformacja - polegająca na manipulowaniu modelem przez przesunięcia, obroty i obliczanie równań perspektywy w celu przedstawienia wyświetlanego obiektu;
- obcinanie - polegające na obcinaniu wymiarów rysunku do rozmiarów powierzchni, na której zostanie przedstawiony;
- wyświetlanie - obiekt jest rysowany na urządzeniu wyjściowym.

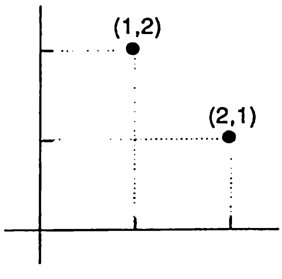


Rys. 5.2. Proces graficzny

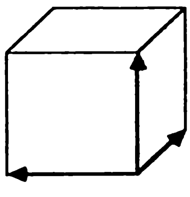
Dokładne określenie, w którym etapie procesu rysowania nastąpi obcinanie obiektu jest ważne i zależy od właściwości posiadanego urządzenia wyjściowego. W niektórych przypadkach należy *obcinać* przed przesłaniem informacji o rysunku do urządzenia wyświetlającego. W innych, można przesłać informację nieobciętą, a urządzenie wyjściowe samo dokona obcięcia. Mimo, że z punktu widzenia skończonego produktu wynik będzie ten sam, to jednak szybkość przetwarzania będzie różna. Urządzenie, które ma możliwość obcinania, ma szansę być znacznie szybsze.

6. Modelowanie - układy współrzędnych

Podstawą każdej metody modelowania geometrycznego jest układ współrzędnych, określający za pomocą zbioru liczb położenie obiektu w przestrzeni dwu- lub trójwymiarowej. Punkt w przestrzeni dwuwymiarowej może być zdefiniowany w sposób jednoznaczny za pomocą pary liczb, a w przestrzeni trójwymiarowej za pomocą trzech liczb. W celu uniknięcia wątpliwości liczby te, muszą być podawane w ściśle określonej kolejności; jak widać na rys. 6.1 punkt o współrzędnych (1,2) ma inne położenie niż punkt (2,1). Układ współrzędnych jest parą lub trójką osi ortogonalnych (zależnie od reprezentacji obiektów dwuwymiarowych lub trójwymiarowych), tzn. ustawionych pod kątem prostym w stosunku do siebie. W przypadku trzech wymiarów osie tworzą róg sześcianu (rys. 6.2).



Rys. 6.1. Układ współrzędnych

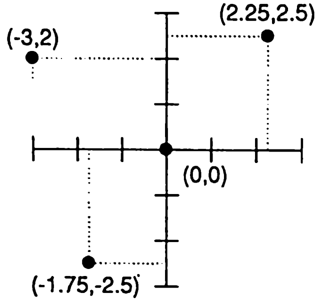


Rys. 6.2. Osie ortogonalne

Istnieje wiele różnych układów współrzędnych, używanych do opisu obiektów. W książce został przyjęty do opisu i rozważań jeden - powszechnie znany pod nazwą *układu kartezjańskiego*. Nazwa układu wywodzi się od nazwiska fracuskiego filozofa i matematyka Descartesa (w Polsce nazywanego Kartezjuszem), żyjącego w latach 1596 - 1650. Jego zasługą było ustalenie zasad geometrii w układzie współrzędnych.

6.1. Współrzędne w układzie dwuwymiarowym

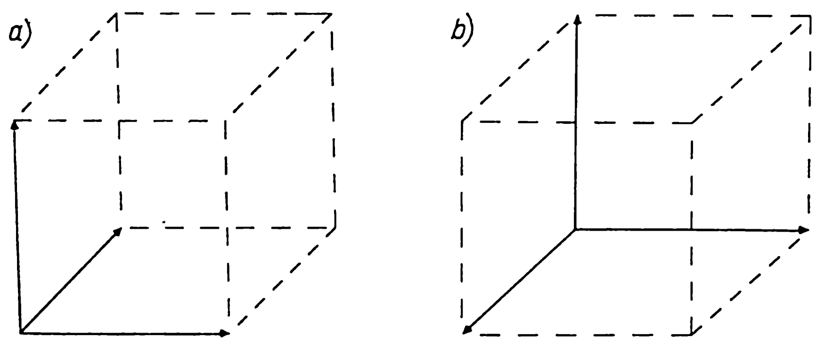
W ~~kartezjańskim~~, dwuwymiarowym układzie współrzędnych używa się dwóch osi. Położenie punktu w tym układzie jest określane parą liczb. Pierwsza liczba odnosi się do osi poziomej, przy czym liczbom wzrastającym odpowiada przesuwanie się w prawo wzdłuż osi. Druga liczba odnosi się do osi pionowej, przy czym liczbom wzrastającym odpowiada przesuwanie się w górę wzdłuż tej osi. Ten układ współrzędnych będzie stosowany w książce. Punkt (0,0) oznacza początek układu współrzędnych i jest punktem przecięcia się osi. Współrzędne punktu mogą przyjmować wartości dodatnie lub ujemne, przy czym mogą być liczbami całkowitymi lub ułamkowymi (rys. 6.3).



Rys. 6.3. Współrzędne na płaszczyźnie

6.2. Współrzędne w układzie trójwymiarowym

W trójwymiarowym układzie współrzędnych są trzy osie: dwie takie same, jak w układzie dwuwymiarowym i trzecia prostopadła do pozostałych. Pierwsze dwie współrzędne są przedstawione tak samo, jak w układzie dwuwymiarowym, trzecia zaś może być przedstawiona na dwa sposoby. Jeśli wyobrazimy sobie, że dwuwymiarowy układ współrzędnych leży w płaszczyźnie stronicy, to trzecia oś jest do niej prostopadła. Liczby określające wartość trzeciej współrzędnej mogą rosnąć w dół, tworząc tzw. *lewoskrętny układ* współrzędnych (rys. 6.4a) lub w górę , tworząc tzw. *prawoskrętny*



Rys. 6.4. Układ współrzędnych: a) lewoskrętny, b) prawoskrętny

układ współrzędnych (rys. 6.4b). Tak więc w układzie lewoskrętnym współrzędne (2,3,1) oznaczają punkt położony 2 jednostki na prawo od początku układu, 3 jednostki w górę i 1 jednostkę w głąb (dalej od oka). Punkt (0,0,0) jest początkiem układu współrzędnych w obu systemach

Mimo, że programy korzystające z lewoskrętnego układu współrzędnych można łatwo przerobić na stosujące prawoskrętny układ współrzędnych, to jednak należy konsekwentnie używać jednego. W przeciwnym razie można doprowadzić do wielkiego bałaganu. W książce jest stosowany lewoskrętny układ współrzędnych.

6.3. Rodzaje współrzędnych

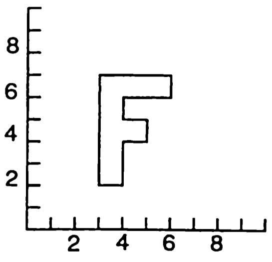
Do zastosowań w grafice komputerowej są potrzebne dwa rodzaje współrzędnych:

- współrzędne świata, używane do definiowania elementów i obiektów przetwarzanych na obraz; są to współrzędne używane do modelowania świata rzeczywistego,
- współrzędne urządzenia wyświetlającego, używane do reprezentacji; są to współrzędne używane do oglądania modelu.

Z powodu używanych różnych jednostek, współrzędne te rzadko są takie same - nawet w przypadku obiektów płaskich.

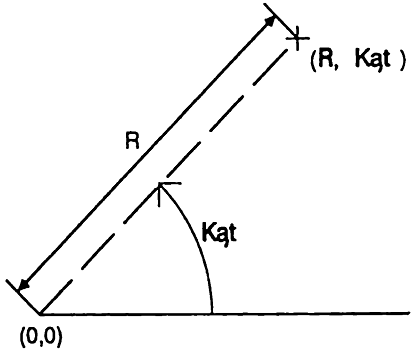
Współrzędne świata

W przypadku programów użytkowych współrzędne są wyrażane w jednostkach świata rzeczywistego. Są stosowane współrzędne w układzie dwuwymiarowym i trójwymiarowym. Na przykład, dyrektor przedsiębiorstwa do wykresu kosztów konserwacji urządzeń w funkcji czasu użyje układu dwuwymiarowego. Na jednej osi będą koszty w złotówkach, na drugiej czas w latach. Naukowiec użyje jednostek takich jak wolty, nanosekundy, parseki itp.; architekt będzie mierzył obiekty w metrach lub milimetrach w układzie trójwymiarowym. Czasami, jak w przypadku litery F, pokazanej na rys. 6.5, można użyć specjalnych jednostek, wymyślonych na taką okazję. W modelowaniu świata rzeczywistego konieczne jest stosowanie ujemnych i dodatnich wartości liczb oraz dwu- i trójwymiarowego układu współrzędnych, lewo- lub prawoskrętnego.



Rys. 6.5. Litera F

Można stosować nie tylko specjalne jednostki, ale także inne postacie układu współrzędnych, np. *układ współrzędnych biegunowych,* w którym położenie punktu jest

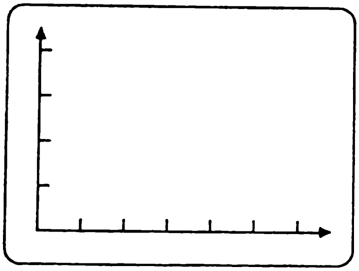


Rys. 6.6. Układ współrzędnych biegunowych: (położenie jest określone za pomocą kąta i odległości)

zdefiniowane za pomocą kąta i odległości (rys. 6.6). Ta postać układu współrzędnych jest wygodna w wielu zastosowaniach. W książce przyjęto, że do celów grafiki komputerowej takie współrzędne zostały przekształcone przez program na współrzędne kartezjańskie świata.

Współrzędne urządzenia

Współrzędne urządzenia odnoszą się do rozmiarów powierzchni, na której jest wyświetlany obraz. W wielu urządzeniach wyświetlających przyjmuje się, że współrzędne mają początek w lewym dolnym rogu i dodatnie półosie wskazują kierunek w górę i w prawo (rys. 6.7). W innych urządzeniach początek układu współrzędnych znajduje się w lewym górnym rogu i półosie dodatnie wskazują kierunek w dół i w prawo. Jeszcze rzadsze są urządzenia mające początek układu współrzędnych w środku ekranu. Przed rozpoczęciem programowania należy zapoznać się z konwencją przyjętą w urządzeniu użytkownika, tak aby uzyskać zamierzone wyniki.



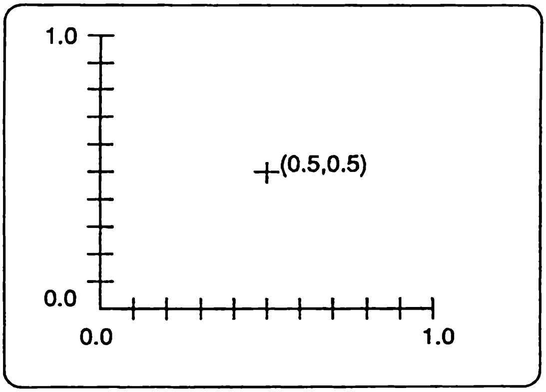
Rys. 6.7. Typowa postać współrzędnych ekranowych (wartości wzrastają w górę i w prawo)

Obecnie, jednostki współrzędnych w urządzeniach są takie, jakie fantazja podsunęła producentowi. Mogą to być cale lub centymetry, częściej jednak są to jednostki całkowicie arbitralne w zakresie takim jak: od 0 do 130, od 0 do 100, od 0 do 279 lub od 0 do 1023 itp. Ostatnio przyjęty w grafice komputerowej **standard GKS** wymaga, aby współrzędne były podawane w rzeczywistych jednostkach - mierzonych

w metrach, a początek układu znajdował się w lewym dolnym rogu. Jednym z zadań w oprogramowaniu grafiki komputerowej jest przeliczanie współrzędnych świata na współrzędne urządzenia tak, aby użytkownik nie musiał sam zajmować się tymi szczegółami. W dobrze opracowanym programie graficznym zadanie to jest realizowane przez trzy powiązane koncepcje: znormalizowane współrzędne urządzenia, okna i pola prezentacji.

6.4. Znormalizowane współrzędne urządzenia

Przy użyciu znormalizownych współrzędnych urządzenia (ang. *normalised device coordinates* - NDCS) przyjmuje się, że powierzchnia robocza rozciąga się od 0.0 do 1.0 w dwóch kierunkach, niezależnie od faktycznego jej kształtu i rozmiaru. Początek takiego dwuwymiarowego układu współrzędnych został ustawiony w lewym dolnym rogu powierzchni roboczej (rys. 6.8). Znormalizowane współrzędne urządzenia są używane jako jednostki pośrednie między współrzędnymi świata a współrzędnymi urządzenia, w celu odseparowania procesu obliczania obrazu od jego wyświetlania. Można zatem



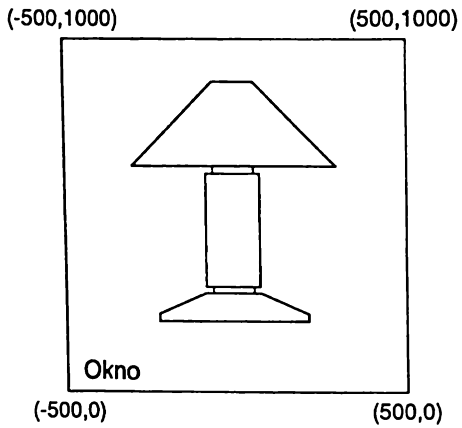
Rys. 6.8. Znormalizowane współrzędne urządzenia (obszar roboczy zacieniowany)

obliczać obraz, tak jakby miał być wyświetlany przez umowne urządzenie o rozmiarach 1.0×1.0, po czym należy wyskalować wyniki do rozmiarów odpowiadających zastosowanemu urządzeniu wyświetlającemu. Dodatkową zaletą tej metody prezentacji jest to, że tak wyskalowany i zapamiętany obraz w postaci pliku dyskowego, może być później przedstawiony na dowolnym innym urządzeniu wyjściowym.

6.5. Okna

Przed rozpoczęciem procesu tworzenia rysunku, należy zdecydować, jaki wycinek świata realnego ma on przedstawiać. Na przykład projektant lamp stołowych

wykonujący rysunek swego produktu może uznać, że jedyna część świata, jaką potrzebuje rozważać ma postać sześcianu o boku 1 metra. Wynika to z założenia, że lampy, które chce projektować mają nie więcej niż 1 m szerokości i 1 m wysokości. Innymi słowy *okno*, którym się posłuży ma wymiary: 1 m wysokości i 1 m szerokości. Ponieważ lampa jest symetryczna względem osi pionowej, projektant prawdopodobnie zdefiniuje zakres okna: w poziomie od –500 mm do +500 mm, a w pionie od 0 do +1000 mm (rys. 6.9).



Rys. 6.9. Lampa stołowa

Natomiast historyk, który będzie badał zmiany zaludnienia od 85 r. p.n.e. do 2085 r., zechce badania te przedstawić na wykresie o oknie rozciągającym się w poziomie od –85 do +2085, a w pionie powiedzmy od –20 000 do +1 000 000. Jeżeli zechce przedstawić wyniki badań tylko w latach od odkrycia Ameryki do pierwszej wojny światowej, to okno będzie rozciągać się w poziomie od +1492 do +1914.

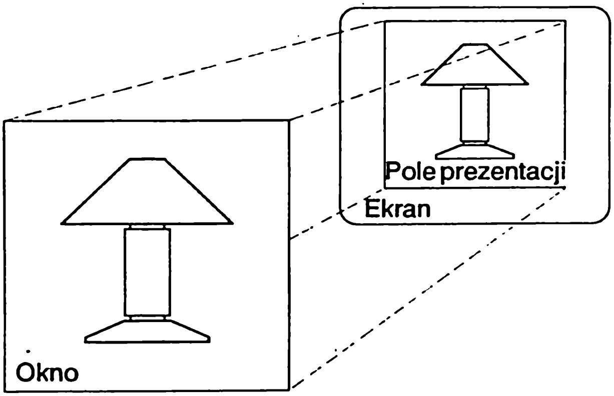
Jak widać z tych przykładów, okno jest związane z przestrzenią dwuwymiarową, może być symetryczne i mieć dowolne proporcje i wymiary. Ponadto okno może być większe lub mniejsze od przedstawianego przedmiotu. Jeśli jest mniejsze, to jedynie część świata mieszcząca się w oknie będzie wyświetlona. Przez zmianę rozmiarów okna można zmieniać skalę i proporcje obrazu. Dwukrotne zwiększenie rozmiarów okna zmniejsza dwukrotnie skalę; dwukrotne zmniejszenie rozmiarów okna zwiększa dwukrotnie skalę. Przy zwiększaniu skali część obrazu wychodzi poza powierzchnię roboczą. Może to spowodować kłopoty, jeżeli urządzenie wyświetlające nie poradzi sobie samo z takim problemem albo program nie podejmie zapobiegawczego działania.

Jedna z metod zapobiegania może polegać na przeszukaniu wszystkich danych i pominięciu tych, które wychodzą poza okno. Taka procedura będzie skuteczna np. w przypadku zadania historyka - przetwarzanie rozpocznie się od danych z roku 1492 i skończy się na danych roku 1914. Metoda nie jest skuteczna w każdym przypadku. Na przykład, do wyświetlenia perspektywicznego widoku obiektu są potrzebne

wszystkie dane o jego geometrii, nawet jeśli tylko część widoku będzie wyświetlona. Należy zatem określić cały rysunek i obciąć tę część, która wychodzi poza okno. Jest to prosty, ale nie trywialny proces matematyczny. Linie, które przecinają narożnik okna muszą być obcięte z obydwu końców i nie można z góry wybrać linii, które mają być obcięte. Wszystkie linie muszą podlegać procesowi obcinania. Niektóre urządzenia wyświetlające są w stanie wykonać ten proces samodzielnie. Użytkownik definiuje jedynie okno i przesyła do urządzenia cały rysunek; urządzenie wyświetla tylko tę część, która jest wybrana przez niego.

6.6. Pole prezentacji

W dyskusji o oknach milcząco przyjęto, że cała powierzchnia robocza urządzenia wyświetlającego będzie użyta do przedstawienia rysunku. W rzeczywistości wykorzystuje się jedynie część tej powierzchni i pozostawia się miejsce na tekst lub na przedstawienie innych rysunków. Na ekranie można zdefiniować *pole prezentacji*. Jest to zadany prostokątny obszar na powierzchni roboczej, w którym zostaną określone znormalizowane współrzędne urządzenia i na który zostanie odwzorowane okno (rys. 6.10).



Rys. 6.10. Odwzorowanie okna na pole prezentacji

Podstawowe warunki definiowania pola prezentacji:

- proporcje okna i pola prezentacji muszą do siebie pasować, w przeciwnym przypadku może wystąpić zniekształcenie obrazu;
- pole prezentacji (w większości systemów graficznych) może być tylko mniejsze od powierzchni roboczej, w przeciwnym wypadku mogą wystąpić nieoczekiwane wyniki;
- pola prezentacji w razie potrzeby mogą zachodzić na siebie na powierzchni roboczej.

ZADANIA

6.1. Zauważyliśmy (rys. 6.1), że porządek podawania liczb określających współrzędne jest ważny i punkt (X,Y) jest na ogół różny od punktu (Y,X). Jeśli odwrócimy współrzędne w opisie obiektu, to wystąpi interesujący efekt. Aby zobaczyć ten efekt, wykreśl na papierze kratkowanym linię według następującej listy współrzędnych, następnie odwróć kolejność liczb w parach i znów wykreśl rysunek. Co za efekt osiągnęliśmy? (Wskazówka: narysuj na wykresie linię pod kątem 45°, przechodzącą przez początek układu współrzędnych). Oto lista współrzędnych punktów: (1,5), (3,3), (2,3), (3,4), (3,3).

6.2. Używając tej samej listy punktów zamień X na –X. Co stanie się z rysunkiem? Wypróbuj to samo przy zamianie Y na –Y. Potem wykonaj te zamiany jednocześnie. Czyżbyśmy odkryli metodę uzyskiwania odbić lustrzanych rysunku?

6.3. W jakich przypadkach współrzędne biegunowe (rys. 6.6) będą bardziej użyteczne niż kartezjańskie? Używając współrzędnych biegunowych wykreśl następujące punkty: (0,8), (30,8), (60,7), (90,6), (120,5), (150,4), (180,3), (210,2), (240,1), (270,0). Pamiętaj, że w tym przypadku pierwsza liczba to kąt w stopniach, a druga to odległość od początku układu. Co trzeba zrobić, aby wykreślić rysunek bardziej podobny do spirali?

6.4. Co się stanie, jeśli opiszesz obiekt trójwymiarowy w lewoskrętnym układzie współrzędnych a program stosuje układ prawoskrętny? Jak zmienić jedno przedstawienie na drugie? Przekonaj się, że przez pomnożenie współrzędnej Z przez –1 zmieniamy układ prawoskrętny na lewoskrętny (i odwrotnie). Zauważ, że proces ten zmienia wszystkie ujemne wartości Z na dodatnie, a dodatnie na ujemne. Czy wynik jest taki sam jak obrócenie obiektu wokół osi Y? W jaki sposób jest inny?

6.5. Określ współrzędne urządzenia swojego systemu graficznego. Informację tę znajdziesz w opisie użytkowym i będzie ona podana w pikselach. Niektóre systemy pozwolą ci wybrać zakres rozdzielczości. Wypisz możliwości swojego systemu.

6.6. Przyjmij, że jesteś architektem i chcesz narysować plan budynku o wymiarach 40 m × 60 m. Załóżmy, że rysunek ma być przedstawiony na ekranie o współrzędnych urządzenia równych 240 × 180 pikseli. Opracuj poprawną procedurę skalowania obiektu, tak aby rysunek był możliwie największy. Pamiętaj o konieczności zachowania proporcji.

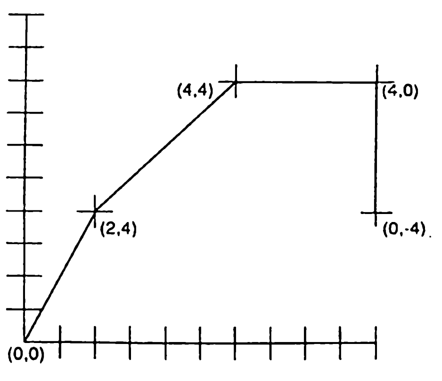
6.7. Powtórz poprzednie zadanie dla obiektu o rozmiarze 600 × 600 m. Pamiętaj o potrzebie uzyskania możliwie dużego rysunku przy zachowaniu proporcji.

7. Modelowanie dwuwymiarowych obiektów złożonych z odcinków linii prostych

Obiekt dwuwymiarowy złożony z odcinków linii prostych, tak jak litera F pokazana na rys. 6.5, można opisać za pomocą listy współrzędnych wszystkich jego wierzchołków. Współrzędne są opisane w kolejności ich pojawiania się przy rysowaniu obwodu obiektu. Z rysunku można odczytać następujące współrzędne, tworzące listę:

(3,2), (3,7), (6,7), (6,6), (4,6), (4,5), (5,5), (5,4), (4,4), (4,2), (3,2)

Literę można łatwo zrekonstruować, wykreślając linie między kolejnymi punktami począwszy od pierwszej pary współrzędnych z tej listy. Nie jest to jedyna metoda opisania i wykreślenia tego rysunku. Przy zachowanej liście współrzędnych, można wystartować od innej pary i posuwać się w kierunku przeciwnym do kierunku ruchu wskazówek zegara. Dodatkowo, można użyć innych jednostek pomiarowych lub ustawić literę w innym miejscu układu współrzędnych. Ponieważ współrzędne z listy są odniesione do układu o ustalonym położeniu środka, to noszą one nazwę *współrzędnych absolutnych*. Czasami wygodniejsze w użyciu są *współrzędne względne*, dla których przyjmuje się, że



Rys. 7.1. Współrzędne względne (każdy punkt linii jest określony w stosunku do swego poprzednika)

początek każdego odcinka (za wyjątkiem pierwszego) znajduje się w końcu poprzedniego, tak jak to pokazano na rys. 7.1. Współrzędne względne mają swoje zalety, ale nie mogą być tak łatwo przetwarzane jak ich absolutne odpowiedniki.

Użycie listy współrzędnych w programie wymaga więcej informacji; np. do komputera należy podać do czego służą te liczby i ile będzie zadanych par. Można to podać bezpośrednio albo za pomocą jakiegoś kodu sygnalizującego koniec tej listy, np. za pomocą pary dużych liczb 99999,99999 - takich, których nie można pomylić z normalnymi współrzędnymi. Jeżeli zostaną użyte współrzędne absolutne, to można zapamiętać opis punktów w dwuwymiarowej tablicy (tabl. 7.1).

Tablica 7.1.

```
/* Załaduj tablicę F(N,2) opisującą współrzędne litery F */
Read N /* Liczba punktów */
Dim F(N,2)
For J = 1 to N
  Read F(J,1),F(J,2)
Next J
Data 11
Data 3,2,3,7,6,7,6,6,4,6,4,5
Data 5,5,5,4,4,4,4,2,3,2
```

Taki opis obiektów dwuwymiarowych jest bardzo dobry, jak bowiem wiadomo już od czasów Euklidesa (żył około 2300 lat temu), przez dwa punkty można poprowadzić tylko jedną linię prostą. Dodatkową zaletą jest powiązanie wszystkich współrzędnych X i Y punktów linii za pomocą prostej zależności:

$$AX + BY + C = 0$$

Oznacza to, że jeśli są znane wartości współczynników A, B i C, to można obliczyć wartości współrzędnych X i Y i odwrotnie. Ponadto, takie sformułowanie tej zależności pozwala na obliczenie kąta nachylenia linii prostej lub punktu przecięcia się dwóch prostych (Dodatek 1).

Jeżeli dane są dwa punkty o współrzędnych (X1,Y1) i (X2,Y2), to współczynniki A, B i C linii prostej łączącej oba punkty, można obliczyć z zależności:

$$A = Y2 - Y1$$
$$B = -(X2 - X1)$$
$$C = -(X1 * A + Y1 * B)$$

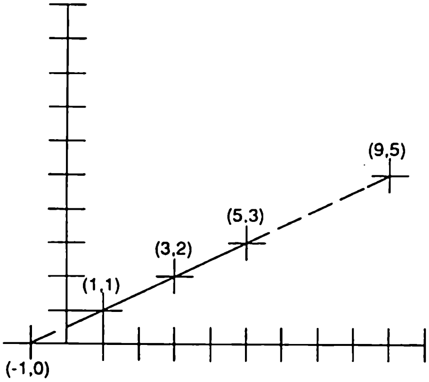
Zależność $AX + BY + C = 0$ jest znana pod nazwą *ogólnego równania prostej*. Współczynniki A, B i C dla prostej przechodzącej przez punkty (1,1) i (5,3) można obliczyć według podanych zależności następująco:

$$A = 3 - 1 = 2$$
$$B = -(5 - 1) = -4$$
$$C = -(1 * 2 + 1 * (-4)) = 2.$$

Po podstawieniu do AX + BY + C = 0, obliczonych wartości współczynników A, B, C otrzymujemy równanie tej prostej w postaci 2X – 4Y + 2 = 0 (lub w prostszej postaci X – 2Y + 1 = 0). Przez podstawienie do

X = 2Y – 1

sprawdzamy, że jeśli Y = 1, to X = 1 oraz jeśli Y = 3, to X = 5, czego należało oczekiwać. Dodatkowo, jeśli Y = 2, to X = 3, czyli punkt (3,2) leży na tej prostej (rys. 7.2).



Rys. 7.2. Linia wykreślona przez zadane punkty

Ogólne równanie prostej powstało przy założeniu, że prosta ma nieskończoną długość. Wychodząc z tego założenia stwierdzamy, że prosta rozciąga się poza punkty (X1,Y1) i (X2,Y2) w obydwu kierunkach w nieskończoność. Można na przykład sprawdzić, że prosta ta przechodzi przez punkty (–1,0) i (9,5), nawet jeśli te punkty leżą poza końcami rozważanego odcinka.

Nieskończona długość prostej jest kłopotliwa w grafice komputerowej i jeśli należy ograniczyć opis tylko do części leżącej między zadanymi punktami, to stosuje się inny opis prostej - tzw. *opis parametryczny*. Zgodnie z tym opisem, każdy punkt leżący na odcinku łączącym dwa punkty końcowe (X1,Y1) i (X2,Y2) ma współrzędne:

X = X1 + t * (X2 – X1)
Y = Y1 + t * (Y2 – Y1)

przy czym, punkty odcinka leżą między punktami końcowymi wówczas, gdy wartość *t* zmienia się między 0 a 1. Tak więc wszystkie punkty odcinka łączącego punkty (1,1) i (5,3) będą miały współrzędne

X = 1 + t * (5 – 1) = 1 + 4 * t
Y = 1 + t * (3 – 1) = 1 + 2 * t

przy założeniu, że *t* zawiera się między 0 a 1

Dla t = 0 punkt ma współrzędne (1,1) a dla t = 1 - współrzędne (5,3), czego należało oczekiwać. Dla t = 0.5 współrzędne wynoszą (3,2) i punkt leży w połowie między punktami końcowymi. Przez proste przekształcenie otrzymujemy:

t = (X – 1)/4 oraz
t = (Y – 1)/2

Korzystając z tych zależności możemy sprawdzić, czy punkt (–1,0) leży na tym odcinku. W tym przypadku:

$t = (-1 - 1)/4 = -0.5$ oraz

$t = (0 - 1)/2 = -0.5$.

Ponieważ –0.5 jest mniejsze niż 0, to punkt o współrzędnych (–1,0) nie leży na odcinku lecz na jego przedłużeniu. Jeśli otrzymane wartości dla t są różne, to punkt nie leży na prostej - niezależnie od tego czy wartości są z zakresu między 0 a 1.

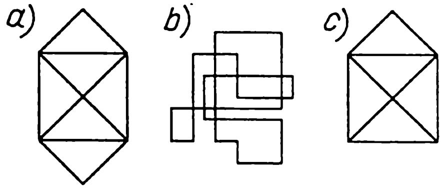
Ogólna i parametryczna postać równania są bardzo użyteczne w grafice komputerowej i czytelnik powinien je sobie dobrze przyswoić.

Inne reprezentacje

Litera F narysowana blokowo, może być całkiem wygodnie przedstawiona jako ciągła lista współrzędnych w tablicy. Wynika to ze sposobu narysowania jej jako prostej, zamkniętej figury geometrycznej, w której koniec każdego odcinka jest początkiem następnego. Inne, bardziej skomplikowane, figury można również przedstawić w postaci linii ciągłej (taką figurę określamy mianem jednokreślnej - przyp. tłum.), pod warunkiem, że spełniają jeden z warunków:

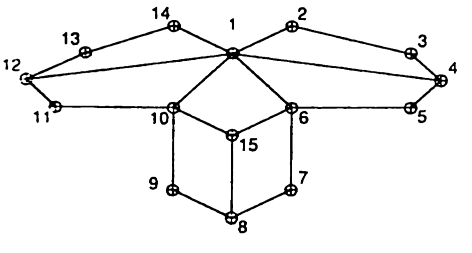
- wszystkie wierzchołki mają parzystą liczbę odcinków wychodzących z nich,
- są tylko dwa wierzchołki, z których wychodzi nieparzysta liczba odcinków.

Jeśli pierwszy warunek jest spełniony, to każdy wierzchołek figury może być wybrany jako początek drogi. W drugim przypadku jako początek trzeba wybrać jeden z wierzchołków z nieparzystą liczbą boków. Z tych reguł wynika jasno, że obiekty (a) i (b) przedstawione na rys. 7.3 są jednokreślne, a obiekt (c) nie jest.



Rys. 7.3. Tylko obiekty a) i b) mogą być jednokreślne

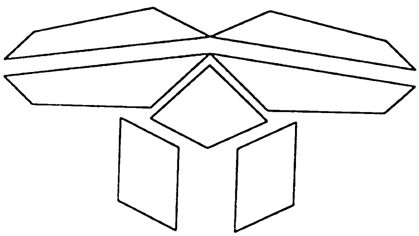
Jeśli nie jest spełniony warunek jednokreślności albo trzeba narysować obiekt bardziej skomplikowany, to można zastosować inną metodę opisu. Na przykład, do opisania obiektu przedstawionego na rys. 7.4 (który może być znakiem firmowym), nie



Rys. 7.4. Złożony obiekt (znak firmowy) niemożliwy do wykreślenia za pomocą jednej linii

wystarczy pojedyncza lista, gdyż figura ma cztery wierzchołki z nieparzystą liczbą boków. Do opisania takiego obiektu można zastosować dwie metody.

Pierwsza metoda polega na podziale obiektu na prostsze figury (jednokreślne) i zapamiętaniu odpowiednich list w oddzielnych tablicach. Przykładowy podział figury na części prostsze pokazano na rys. 7.5. Niektóre z wypisanych współrzędnych



Rys. 7.5. Jeden z możliwych podziałów obiektu z rys. 7.4

punktów, części składowych, pojawiają się dwa lub trzy razy. W tym przykładzie jest tylko 15 punktów, ale 23 pary współrzędnych. W skomplikowanych przypadkach występuje częstsze powtarzanie par współrzędnych, co wymaga zwiększonej pojemności pamięci i prowadzi do błędów przy wpisywaniu i redagowaniu zwiększonej liczby współrzędnych. Jednakże podstawową wadą opisu, w którym występuje powielanie tych samych punktów wiele razy, jest zwalnianie procesu transformacji.

W drugiej metodzie każdemu wierzchołkowi nadaje się numer i zapisuje się jego współrzędne. Następnie sporządza się listę numerów punktów w porządku, w jakim będą łączone liniami. Lista wierzchołków jest przechowywana (jak poprzednio) w tablicy dwuwymiarowej, a lista linii - w tablicy jednowymiarowej. Lista może być zorganizowana tak, że zawiera zarówno indeksy punktu jak i wyliczenie numerów punktów. Takie listy są pokazane w tabl. 7.2.

Zdarzają się jednak przypadki, że nawet taki opis jest niewystarczający. Przykładowo czasem są potrzebne oddzielne listy linii, dla poszczególnych wielokątów, np. wówczas, gdy chcemy, aby komputer zamalował je na różne kolory albo zacieniował. W takim przypadku ta sama lista punktów wystarcza całkowicie, ale lista linii musi być taka, jak pokazano w tabl. 7.3. Pojedyncza liczba w wierszu pokazuje, ile jest wierzchołków w następnym wierszu tabeli. Tak się akurat zdarzyło, że wszystkie wielokąty w omawianym przykładzie są czworokątami i są zamknięte (pierwszy wierzchołek jest jednocześnie ostatnim). Pozwala to zredukować tę listę do postaci, która również zostanie poprawnie zdekodowana (tabl. 7.4). W ogólnym przypadku nie można mieć jednak pewności, że zawsze wystąpią tylko czworokąty i że będą zamknięte. Tak więc bezpieczniej jest napisać podprogram opisu tak, aby posługiwał się pełnymi danymi - nawet jeśli zajmuje to więcej miejsca. Wykaz wierzchołków i lista linii dostatecznie dobrze opisują geometrię figur płaskich. Do określenia typów linii, kolorów i cieniowania jest potrzebna większa ilość informacji.

Tablica 7.2.

Lista współrzędnych punktów

Punkt	X	Y
1	0	2
2	2	3
3	6	2
4	7	1
5	6	0
6	2	0
7	2	-3
8	0	-4
9	-2	-3
10	-2	0
11	-6	0
12	-7	1
13	-6	2
14	-2	3
15	0	-1

Lista linii

5
1,12,13,14,1
14
1,2,3,4,5,6,7,8,9,10,11,12,1,4
6
8,15,10,1,6,15
0

Tablica 7.3.

5
1, 2, 3, 4, 1
5
1. 4, 5, 6, 1,
5
1, 6, 15, 10, 1
5
6, 7, 8, 15, 6
5
15, 8, 9, 10, 15
5
1, 10, 11, 12, 1
5
1, 12, 13, 14, 1
0

Tablica 7.4

1, 2, 3, 4, 1
1, 4, 5, 6, 1
1, 6, 15, 10, 1
6, 7, 8, 15, 6
15, 8, 9, 10, 15
1, 10, 11, 12, 1
1, 12, 13, 14, 1
0

ZADANIA

7.1. Wykaż, przez odręczne narysowanie, że odjęcie 3 od każdej wartości współrzędnej, X litery F, spowoduje jej przesunięcie w lewo o 3 jednostki. Co się stanie, jeśli dodamy 5 do każdej współrzędnej X i Y? Narysuj kilka razy tę literę przy różnych wartościach przesunięcia, aby przekonać się, że jeśli nie zmieniamy

zależności między współrzędnymi, to zmiana wartości współrzędnych powoduje jedynie przesuwanie figury.

7.2. Pomnóż każdą współrzędną z pary przez ustaloną liczbę i narysuj figurę. Co się stało? Zauważmy, że zmienił się nie tylko rozmiar figury, ale także położenie.

7.3. Czy potrafisz wskazać sytuację, w której współrzędne względne będą bardziej użyteczne niż absolutne? Wskazówka: jak zmienić rozmiar figury bez zmiany jej położenia?

7.4. Napisz program obliczania ogólnego równania odcinka łączącego dwa punkty. Program ten będzie w przyszłości przekształcony na podprogram. Użyj go do potwierdzenia następujących wyników:

(0,0) do (5,5) $X - Y = 0$ (to jest $A = 1, B = -1, C = 0$)
(3,1) do (2,7) $6X + Y - 19 = 0$ lub $X + 0.167Y - 3.167 = 0$
(4,6) do (10,6) $-Y + 6 = 0$
(–4.6,–32) do (500,90.92) $122.92X - 504.6Y - 15581.8 = 0$
lub $X - 4.105Y - 126.764 = 0$

7.5. Użyj programu z zadania 7.4 do sprawdzenia następujących wyników:

(0,0) do (3,7) $A = 7, B = -3, C = 0$
(0,0) do (–3,7) $A = 7, B = 3, C = 0$
(0,0) do (–3,–7) $A = -7, B = 3, C = 0$
(0,0) do (3,–7) $A = -7, B = -3, C = 0$

Co możesz wywnioskować z tych wyników? Czy możesz na tej podstawie odpowiedzieć, dlaczego jedynie prosta, która ma dodatni współczynnik przy Y w zadaniu 7.4 przechodzi przez punkty (3,1) i (2,7)?

7.6. Potwierdź, że dla odcinka łączącego punkty (0,0) oraz (3,7), jeśli $X = 1$, to $Y = 2.333$ i jeśli $X = 6$, to $Y = 14$.

7.7. Napisz program (który także zostanie później przekształcony na podprogram) obliczania współczynników parametrycznego równania odcinka dla następujących odcinków:

(0,0) do (5,5) $X = 5t, Y = 5t$
(3,1) do (2,7) $X = 3 - t, Y = 1 + 6t$
(4,6) do (10,6) $X = 4 + 6t, Y = 6$
(–4.6,–32) do (500, 90.92) $X = -4.6 + 504.6t, Y = -32 + 122.92t$

7.8. Użyj wyników poprzedniego zadania do sprawdzenia gdzie linia prosta $X = 150$ przecina się z liną łączącą punkty (–4.6,–32) i (500, 90.92). Powinieneś otrzymać odpowiedź:

$X = 150, Y = 5.66$ (to jest $t = 0.3064$)

7.9. Wykaż, że $t = 1.333$ wówczas, gdy prosta $Y = 9$ przecina linię prostą przechodzącą przez punkty (3,1) i (2,7). Potwierdź, że punkt przecięcia leży poza odcinkiem, ponieważ $t > 1$.

8. Modelowanie dwuwymiarowych obiektów złożonych z odcinków linii krzywych

Wiemy, że przez dwa różne punkty można narysować tylko jedną linię prostą, a dowolny odcinek linii prostej można opisać podając współrzędne jego końców. Opis obiektów złożonych z linii krzywych jest całkiem inny. Ponieważ przez dwa punkty można poprowadzić dowolną liczbę linii krzywych, opis musi zawierać więcej informacji.

Za pomocą najczęściej używanych urządzeń graficznych, można rysować linie krzywe jedynie przez złożenie dużej liczby odcinków linii prostych, albo jako zbiór rozjaśnionych pikseli. Sposób ten jest pokazany na rys. 8.1. Przy modelowaniu obiektów krzywoliniowych należy wprowadzić do komputera współrzędne odcinków lub punktów. Dla niektórych figur można je podać bezpośrednio, w postaci listy. Wiele krzywych np. okręgi i elipsy daje się opisać zależnościami, z których można obliczyć współrzędne punktów.

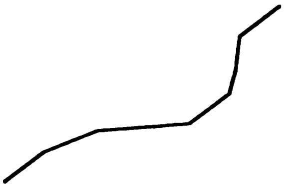
8.1. Okręgi i elipsy

Współrzędne (X,Y) wszystkich punktów znajdujących się na okręgu o środku w punkcie (Xsr, Ysr) i o promieniu równym R są dane zależnościami:

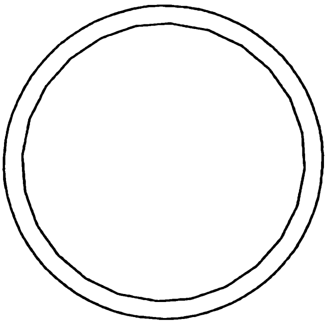
$$X = Xsr + R * \cos(A)$$
$$Y = Ysr + R * \sin(A)$$

w której kąt A zmienia się od 0 do 360° (od 0 do 2 * PI radianów).

Ta zależność opisuje okrąg, jaki można narysować za pomocą cyrkla. Kąt A powinien zmieniać się o nieskończenie małe wartości, ale w większości zastosowań grafiki komputerowej wystarcza, jeśli zmieniać się on dużymi skokami - np. o 5°. Taką metodą uzyskuje się wprawdzie wielokąt o 72 bokach zamiast prawdziwego okręgu, lecz przy dostatecznie małych promieniach jest on łudząco podobny do okręgu (rys. 8.2). Dla bardzo małych promieni nawet mniejsza liczba boków daje zadowalający wynik.



Rys. 8.1. Linia krzywa



Rys. 8.2. Okręgi: jeden złożony z 24, drugi z 72 odcinków

Programy wykorzystujące tę prostą zależność pracują stosunkowo szybko, nawet jeśli muszą w każdym kroku liczyć wartości funkcji sinus i cosinus kąta. Szybkość programu można zwiększyć przez zastosowanie prostego wzoru matematycznego na sinus i cosinus sumy kątów, a mianowicie:

$$\cos(A + DA) = \cos(A) * \cos(DA) - \sin(A) * \sin(DA)$$
$$\sin(A + DA) = \sin(A) * \cos(DA) + \cos(A) * \sin(DA)$$

Korzystając z tej zależności można uniknąć ciągłego obliczania wartości funkcji sinus i cosinus i zastosować zależność rekurencyjną, w której wartości X i Y w jednym punkcie będą użyte do obliczenia współrzędnych w kolejnym punkcie. Oto te zależności:

$$X(I) = Xsr + R * \cos(A)$$
$$Y(I) = Ysr + R * \sin(A)$$
$$X(I + 1) = Xsr + R * \cos(A + DA)$$
$$Y(I + 1) = Ysr + R * \sin(A + DA)$$

Po kilku przekształceniach matematycznych, z drugiej pary zależności i wzorów na sinus i cosinus sumy, otrzymujemy wzory opisujące każdy punkt w funkcji kąta DA:

$$X(I + 1) = Xsr + (X(I) - Xsr) * \cos(DA) - (Y(I) - Ysr) * \sin(DA)$$
$$Y(I + 1) = Ysr + (X(I) - Xsr) * \sin(DA) + (Y(I) - Ysr) * \cos(DA).$$

Dla warunków początkowych: kąt A = 0 i I = 1, otrzymujemy:

$$X(1) = Xsr + R$$
$$Y(1) = Ysr$$

Przy założeniu, że kąt DA ma wartość stałą, można obliczyć jeden raz wartości sinusa i cosinusa kąta DA i używać ich w całym programie. W tablicy 8.1 przedstawiono podprogram szybko obliczający współrzędne okręgu. W programie (zob. komentarze) założono, że komputer oblicza funkcje trygonometryczne kątów podanych w stopniach; większość systemów graficznych pracuje w radianach. W takim przypadku wiersze programu:

$$C = \cos(DA)$$
$$S = \sin(DA)$$

należy zamienić na:

```
F = PI/180 /* zamiana stopni na radiany */
C = cos(DA * F)
S = sin(DA * F)
```

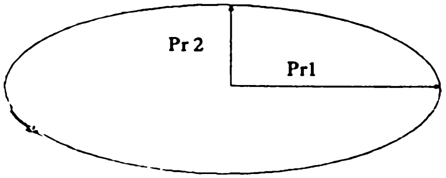
Wielu początkujących programistów zapomina, że obliczenia są prowadzone w radianach, a nie w stopniach. Zwykle łatwiej jest obliczać wszystko wewnątrz programu w radianach, nawet jeśli na wejściu i wyjściu dane są określone w stopniach. Radiany przelicza się na stopnie mnożąc je przez 180/PI. Taki sposób obliczeń jest stosowany w naszych programach. Jeśli liczba boków Num zadawana w programie (tabl. 8.1) jest dostatecznie mała, to podprogram rysuje trójkąt, kwadrat, pięciokąt, itd.

Tablica 8.1. Podprogram kreślenia *okręgu*

```
Subroutine Okrąg(Num, Xsr, Ysr, Promień; N, Outarray)
Integer Num, N
Real Xsr, Ysr, Promień, Outarray
/* Podprogram ładuje do tablicy Outarray współrzędne punktów okręgu o środku
w punkcie (Xsr, Ysr). Okrąg jest aproksymowany za pomocą wielokąta o Num
bokach */
Local Integer J
Local Real DA, C, S, Xdiff, Ydiff
        N = Num + 1
        Dim Outarray(N,2)
        DA = 360/Num   /* kąt liczony w stopniach */
        C = cos(DA)
        S = sin(DA)
        Outarray(1,1) = Xsr + Promień  /* pierwszy punkt */
        Outarray(1,2) = Ysr
        For J = 2 to Num       /* punkty pośrednie */
           Xdiff = Outarray(J-1,1) - Xsr
           Ydiff = Outarray(J-1,2) - Ysr
           Outarray(J,1) = Xsr + Xdiff * C - Ydiff * S
           Outarray(J,2) = Ysr + Xdiff * S + Ydiff * C
        Next J
        Outarray(N,1) = Outarray(1,1)    /* ostatni punkt */
        Outarray(N,2) = Outarray(1,2)
End podprogramu Okrąg
```

W tablicy 8.2 podano podprogram rysowania elipsy (albo okręgu jeśli przyjąć, że jej dwie półosie Pr1 i Pr2 są równe). Elipsę opisuje się, podając dwie osie: mniejszą i większą oraz współrzędne środka (rys. 8.3).

Do przyśpieszenia obliczeń, zwłaszcza jeśli należy rysować dużo okręgów, dobrze jest zapamiętać współrzędne punktów tworzących tzw. *okrąg jednostkowy*, czyli okrąg o promieniu równym 1 i o środku w punkcie (0,0). Narysowanie dowolnego okręgu wymaga przeskalowania okręgu jednostkowego (mnożenie współrzędnych przez



Rys. 8.3. Elipsa i jej półosie: mniejsza i większa

Tablica 8.2 Podprogram kreślenia *elipsy*

```
Subroutine Elipsa(Num, Xsr, Ysr, Pr1, Pr2; N, Outarray)
Integer Num, N
Real Xsr, Ysr, Pr1, Pr2, Outarray
/* Podprogram ładuje do tablicy Outarray współrzędne elipsy o środku w punkcie
(Xsr, Ysr). Elipsa ma większą półoś równą Pr1, mniejszą Pr2 i jest aproksymowana
za pomocą wielokąta o Num bokach */
Local Integer J
Local Real DA, F, C, S
        N = Num + 1
        Dim Outarray(N,2)
        DA = 360/Num          /* kąt liczony w stopniach */
        Outarray(1,1) = Xsr + Pr1   /* pierwszy punkt */
        Outarray(1,2) = Ysr
        F = PI/180
        For J = 2 to Num        /* punkty pośrednie */
          C = cos(DA * F)
          S = sin(DA * F)
          Outarray(J,1) = Xsr + Pr1 * C
          Outarray(J,2) = Ysr + Pr2 * S
        Next J
        Outarray(N,1) = Outarray(1,1)     /* ostatni punkt */
        Outarray(N,2) = Outarray(1,2)
        End podprogramu Elipsa
```

promień) i przesunięcia go do właściwego położenia (proste dodawanie). Podobnie można postąpić z elipsami, skalując oddzielnie współrzędne X i Y (mnożąc je odpowiednio przez Pr1 i Pr2). Podprogram rysowania okręgu jednostkowego jest przedstawiony w tabl. 8.3.

Tablica 8.3. Podprogram kreślenia *okręgu jednostkowego*

```
Subroutine OkrągJednostkowy (Num; N, Outarray)
Integer Num, N
Real Outarray
```

```
/* Podprogram ładuje do tablicy Outarray współrzędne punktów okręgu o środku
w punkcie (0,0) i promieniu równym 1. Okrąg jest aproksymowany za pomocą
wielokąta o Num bokach */
Local Integer J
Local Real DA, F, C, S, Xdiff, Ydiff
        N = Num + 1
        Dim Outarray(N,2)
        F = PI/180
        DA = 360/Num   /* kąt liczony w stopniach */
        C = cos(DA)
        S = sin(DA)
        Outarray(1,1) = 1  /* pierwszy punkt */
        Outarray(1,2) = 0
        For J = 2 to Num   /* punkty pośrednie */
          Xdiff = Outarray(J–1,1)
          Ydiff = Outarray(J–1,2)
          Outarray(J,1) = Xsr + Xdiff * C – Ydiff * S
          Outarray(J,2) = Ysr + Xdiff * S + Ydiff * C
        Next J
        Outarray(N,1) = 1   /* ostatni punkt */
        Outarray(N,2) = 0
End podprogramu OkrągJednostkowy
```

8.2. Łuki okręgu

Do wykreślenia części okręgu (łuku) można użyć zmodyfikowanego podprogramu kreślenia okręgu, zamieszczonego w tabl. 8.4.

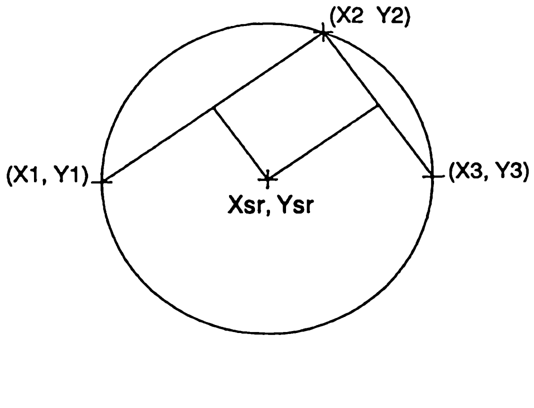
Tablica 8.4. Podprogram kreślenia *łuku*

```
Subroutine Łuk(Num, Xsr, Ysr, KątStartu, KątŁuku, Promień; N, Outarray)
Integer Num, N
Real Xsr, Ysr, KątStartu, KątŁuku, Promień, Outarray
/* Podprogram ładuje do tablicy Outarray współrzędne punktów łuku o środku
w punkcie (Xsr,Ysr) i promieniu równym Promień, poczynając od kąta KątStartu,
kończąc na kącie KątStartu + KątŁuku. Kąty są obliczane w stopniach. Kąty mają
wartości dodatnie w kierunku przeciwnym do ruchu wskazówek zegara. Łuk jest
aproksymowany za pomocą wielokąta o Num bokach */
Local Integer J
Local Real DA, F, SA, AA, EA, C, S, Xdiff, Ydiff
        F = PI / 180
        SA = KątStartu * F
        AA = KątŁuku * F
```

```
        DA = 360 / Num     /* kąt liczony w stopniach */
/* N jest liczbą punktów w trochę dłuższym łuku */
        N = Int((Abs(AA) * 1.01) / DA)
        DA = DA * Sgn(AA) * F  /* aby mieć pewność, że kąt zmienia się */
        Dim Outarray(N,2)      /* w dobrym kierunku */
        C = cos(DA)
        S = sin(DA)
        Outarray(1,1) = Xsr + Promień * cos(SA) /* pierwszy punkt */
        Outarray(1,2) = Ysr + Promień * sin(SA)
        For J = 2 to Num - 1                /* punkty pośrednie */
          Xdiff = Outarray(J-1,1) - Xsr
          Ydiff = Outarray(J-1,2) - Ysr
          Outarray(J,1) = Xsr + Xdiff * C - Ydiff * S
          Outarray(J,2) = Ysr + Xdiff * S + Ydiff * C
        Next J
          EA = SA + AA
        Outarray(N,1) = Xsr + Promień * cos(EA)   /* ostatni punkt */
        Outarray(N,2) = Ysr + Promień * sin(EA)
End podprogramu Łuk
```

8.3. Okręgi przechodzące przez trzy zadane punkty

W podprogramach kreślenia okręgu i łuku założono, że jest znane położenie środka i promień krzywej oraz długość rysowanej krzywej. W praktyce dane te nie są zwykle znane. Jednakże, jeśli są dane współrzędne trzech niepokrywających się punktów, przez które ma przechodzić okrąg, to można obliczyć współrzędne środka i promień tego okręgu za pomocą podprogramu z tabl. 8.5. Obliczenia te wykonuje się wyznaczając punkt przecięcia symetralnych dwóch odcinków: (X1,Y1) – (X2,Y2) oraz (X2,Y2)



Rys. 8.4. Okrąg przechodzący przez trzy punkty

Tablica 8.5. Podprogram *Okrąg3Punkty*

```
Subroutine Okrąg3Punkty(X1,Y1,X2,Y2,X3,Y3,Xsr,Ysr,Promień)
Real X1,Y1,X2,Y2,X3,Y3,Xsr,Ysr,Promień
/* Podprogram oblicza współrzędne środka i promień okręgu przechodzącego
przez trzy punkty (X1,Y1), (X2,Y2), (X3,Y3), podane w tej kolejności */
Local Real D1,D2,D3,D4,XX,YY,F1,F2
        D1 = X2 - X1
        D2 = Y2 - Y1
        D3 = X3 - X1
        D4 = Y3 - Y1
        XX = X1 * X1
        YY = Y1 * Y1
/ * Obliczamy środek odcinków */
        F1 = (XX - X2 * X2 + YY - Y2 * Y2) * 0.5
        F2 = (XX - X3 * X3 + YY - Y3 * Y3) * 0.5
/* Włączamy tutaj nasz test kolinearności punktów */
        Ysr = (D1 * F2 - D3 * F1) / (D3 * D2 - D1 * D4)
        Xsr = -(F1 + D2 * Ysr) / D1
        XX = (X1 - Xsr) * (X1 - Xsr)
        YY = (Y1 - Ysr) * (Y1 - Ysr)
        Promień = sqrt(XX + YY)
End podprogramu Okrąg3Punkty
```

– (X3,Y3) (rys. 8.4). Punkty powinny być podawane w kolejności (X1,Y1), (X2,Y2), (X3,Y3) oraz nie mogą być kolinearne, tzn. nie mogą leżeć na jednej prostej. Opracowanie testu na kolinearność punktów zostawiamy Czytelnikowi. Test ten powinien być włączony do podprogramu. Do obliczenia współrzędnych okręgu przechodzącego przez punkty (10,0), (10,6) i (4,12) należy zastosować program złożony z podprogramów: Okrąg3Punkty i Okrąg (tabl. 8.6).

Tablica 8.6. Fragment programu

```
/* Fragment większego programu do kreślenia okręgu */
        Call Okrąg3Punkty(10,0,10,6,4,12;X,Y,R)
        Call Okrąg(72,X,Y,R;Carray)
/* Koniec fragmentu programu */
```

Alternatywnie można użyć podprogramów Okrąg3Punkty i OkrągJednostkowy (tabl. 8.7). Wersja ta jest prawdopodobnie szybsza w przypadku rysowania wielu

Tablica 8.7. Fragment programu

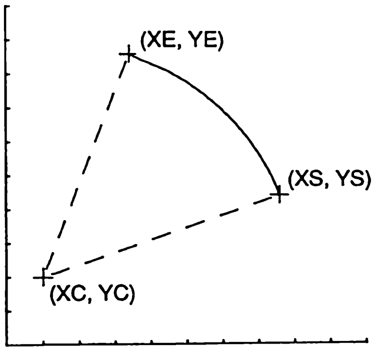
```
/* Fragment większego programu kreślenia okręgu, korzystający z podprogramu
OkrągJednostkowy */
```

```
        Call OkrągJednostkowy(72;Oarray)
        .......................
        Call Okrąg3Punkty(10,0,10,6,4,12;X,Y,R)
/* Przelicz współrzędne okręgu jednostkowego */
        For J = 1 to 72
           Carray(J,1) = Oarray(J,1) * R + X
           Carray(J,2) = Oarray(J,2) * R + Y
        Next J
/* Koniec fragmentu programu */
```

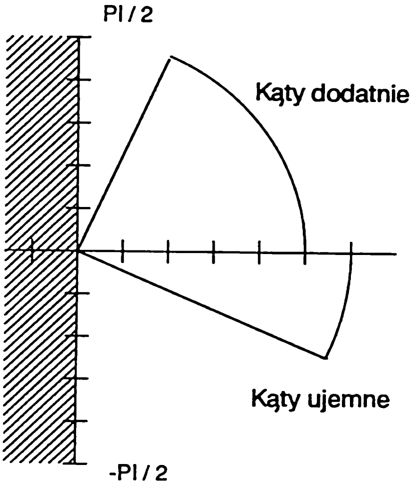
różnych okręgów. Przeliczanie współrzędnych okręgu jednostkowego powinno być wykonywane także za pomocą oddzielnego podprogramu.

Do kreślenia łuków są potrzebne: współrzędne środka i promień, a także kąt początku łuku i jego miara kątowa. Przyjmując, że punkt (XC, YC) oznacza środek łuku, punkt (XS, YS) początek łuku, a punkt (XE, YE) koniec łuku (rys. 8.5), można obliczyć te kąty za pomocą następujących zależności matematycznych:

KątStartu = arctan((YS – YC)/(XS – XC))
KątKońca = arctan((YE – YC)/(XE – XC)).



Rys. 8.5. Kreślenie łuku



Rys. 8.6. Interpretacja funkcji arcus tangens

Funkcja arcus tangens w językach programowania jest często oznaczana ATAN lub ATN i wartość jej jest podawana w radianach. Istnieje pewne utrudnienie wynikające z faktu, że funkcja arcus tangens przyjmuje wartość kąta z zakresu od –PI/2 do +PI/2 lub inaczej od -90° do $+90^{\circ}$ (rys. 8.6). Oznacza to, że proste wywołanie tej funkcji nie

zawsze gwarantuje poprawną odpowiedź. Wynik należy dopasować do aktualnych wymagań. Przyjmując, że obliczamy kąt między osią poziomą, a linią poprowadzoną z punktu (XC, YC) do punktu (X,Y) należy dokonać poprawek pokazanych w tabl. 8.8.

Tablica 8.8. Podprogram *ObliczKąt*

```
Subroutine ObliczKąt(XC,YC,X,Y,Znacznik;A)
Integer Znacznik
Real XC, YC, X, Y, A
/* Podprogram oblicza wartość kąta (w stopniach, jeśli Znacznik = 0
i w radianach, jeśli Znacznik = 1 */
Local Real DX, DY
        If X = XC Then
           If Y > YC Then
              A = PI/2
           Else
              A = 3 * PI/2
           Endif
        Else
           DX = XC - X
           DY = YC - Y
           A = arctan(DY/DX)
        Endif
/* Wykonaj korekcję wartości A */
        If X > XC And Y < YC Then
           A = A + 2 * PI
        Else
           If X < XC Then
              A = A + PI
           Endif
        Endif
/* Jeśli wynik ma być podany w stopniach, to przelicz */
        If Znacznik Then
           A = A * 180 / PI
        Endif
End podprogramu ObliczKąt
```

Dodatkowa korekcja jest również potrzebna wówczas, gdy chcemy podawać kąt w obydwu kierunkach: zgodnie i przeciwnie do kierunku ruchu wskazówek zegara. Należy upewnić się, że kąt początkowy łuku jest większy niż końcowy kąt dla łuków zgodnych z kierunkiem ruchu wskazówek zegara i mniejszy dla łuków o kierunku przeciwnym. Podprogram obliczania środka okręgu, promienia i kątów dla łuku przechodzącego przez trzy punkty jest podany w tabl. 8.9.

Tablica 8.9. Podprogram *Łuk3Punkty*

```
Subroutine Łuk3Punkty(Xp,Yp,Xsr,Ysr,Xkon,Ykon,Znacznik; XC,YC,R,SA,AA)
Integer Znacznik
Real Xp,Yp,Xsr,Ysr,Xkon,Ykon,XC,YC,R,SA,AA
/* Podprogram oblicza położenie środka (XC,YC), promień R, kąt początkowy SA
i miarę kątową AA łuku przechodzącego prze 3 zadane punkty. Wartość kątów
jest podana w stopniach, jeśli Znacznik = 0 i w radianach, jeśli Znacznik = 1 */
Local Integer J
Local Real D1,D2,D3,D4,D9,D0,A,B,C,US,VS,UM,VM,UE,VE,T,F1,F2,F3,
A0,B0,C0,Q,X,Y,CS,SI
/* Najpierw obliczamy środek łuku i promień */
        Call Okrąg3Punkty(Xp,Yp,Xsr,Ysr,Xkon,Ykon;XC,YC,R)
        D1 = Xsr - Xp
        D2 = Ysr - Yp
        D3 = Xkon - Xp
        D4 = Ykon - Yp
        D9 = Xsr - Xkon
        D0 = Ysr - Ykon
        J = 10000 /* współczynnik do testowania */
        A = sqrt(D9 * D9 + D0 * D0)  /* długość A */
        B = sqrt(D1 * D1 + D2 * D2)  /* długość B */
        C = sqrt(D3 * D3 + D4 * D4)  /* długość C */
/* Przesuń punkty do początku układu współrzędnych */
        US = Xp - XC
        VS = Yp - YC
        UE = Xkon - XC
        VE = Ykon - YC
        UM = Xsr - XC
        VM = Ysr - YC
        T = 2 * R * R
        If B > = C Or A > = C Then
           F3 = (T - C * C)/T
           C0 = arcos(F3)
           Q = C0 - 2 * PI

/* Sprawdź kierunek łuku przez obrót punktu początkowego w kierunku ruchu
wskazówek zegara i określ jego położenie  w stosunku do punktu końcowego */
           CS = cos(C0)
           SI = sin(C0)
           X = Int((US * CS - VS * SI) * J)
           Y = Int((US * SI + VS * CS) * J)
           If Int(UE * J) And Int(VE * J) Then
              AA = Q
           Else
```

```
                AA = -Q
             Endif
          Else
             F1 = (T - B * B)/T
             B0 = arcos(F1)
             F2 = (T - A * A)/T
             A0 = arcos(F2)
             Q = A0 + B0
/* Sprawdź kierunek łuku ze środkiem */
             CS = cos(B0)
             SI = sin(B0)
             X = Int((US * CS - VS * SI) * J)
             Y = Int((US * SI + VS * CS) * J)
             If X = Int(UM * J) And Y = Int(VM * J) Then
             AA = Q
             Else
             AA = -Q
             Endif
          Endif
          Call ObliczKąt(XC,YC,Xp,Yp,0;SA)
/* Jeśli chcesz mieć kąt w stopniach, to dokonaj przeliczenia */
          If Znacznik Then
             AA = AA * 180/PI
             SA = SA * 180/PI
          Endif
 End podprogramu Łuk3Punkty
```

Podprogram korzysta z funkcji arcus cosinus. Wówczas, gdy system komputerowy nie ma tej funkcji, należy ją obliczyć za pomocą następującej zależności, korzystając z funkcji arcus tangens:

arcos(X) = PI/2 – arctan(X/sqrt(1 – X * X))

Otrzymany wynik jest wyrażony w radianach, jeśli funkcja arctan także jest przedstawiona w radianach.

W systemie można połączyć podprogramy Okrąg3Punkty i Łuk3Punkty w jeden podprogram pod stosownie wybraną nazwą, np. OkrągInfo. Zależnie od ustawienia pierwszego parametru w wyniku otrzymuje się szukane parametry okręgu lub łuku. Tak więc wywołanie:

OkrągInfo(0,10,0,10,6,4,12,Zn;XC,YC,R,SA,EA)

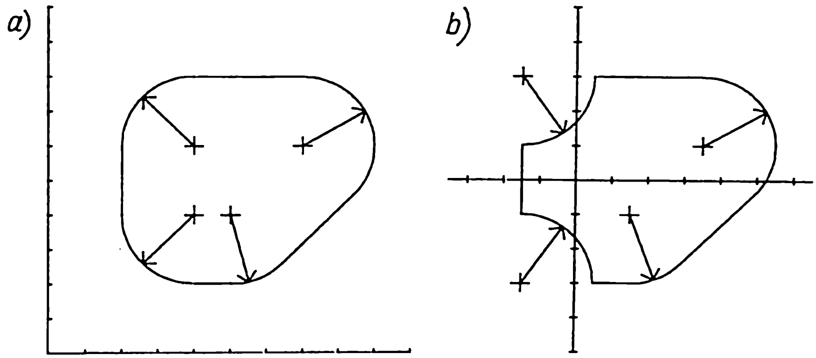
(pierwszy parametr jest zerem) uruchamia podprogram, w wyniku którego otrzymujemy dane o okręgu przechodzącym przez punkty (10,0), (10,6) i (4,12). Wywołanie:

OkrągInfo(1,10,0,10,6,4,12,Zn;XC,YC,R,SA,EA)

(pierwszy parametr jest jedynką) uruchamia podprogram, w wyniku którego otrzymujemy dane o łuku zaczynającym się w punkcie (10,0) i kończącym się w punkcie (4,12). Korzystając z obliczonych parametrów można wywołać odpowiedni podprogram kreślenia okręgu lub łuku.

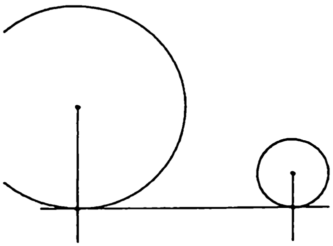
8.4. Styczne i łuki

Rysowanie obrazu złożonego z łuków i odcinków linii prostych (rys. 8.7), sprowadza się do obliczenia współrzędnych środków okręgów i punktów przejścia łuków w linie proste. Jest to szczególnie ważne do uzyskania efektu płynnego przejścia linii i łuków, takiego jak pokazano na rys. 8.7a. Warunkiem osiągnięcia płynności przejścia jest

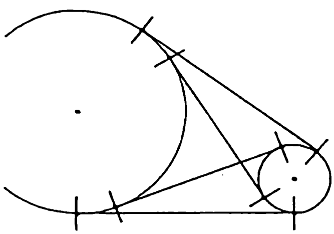


Rys. 8.7. Rysunek złożony z łuków i odcinków; a) odcinki i łuki 1 b) odcinki i łuki 2

styczność linii do łuków, tzn. linia prosta i promień łuku muszą przecinać się pod kątem prostym (rys. 8.8). Punkty przecięcia nazywamy *punktami styczności*. Stosując twierdzenia i zależności geometryczne można obliczyć współrzędne punktów



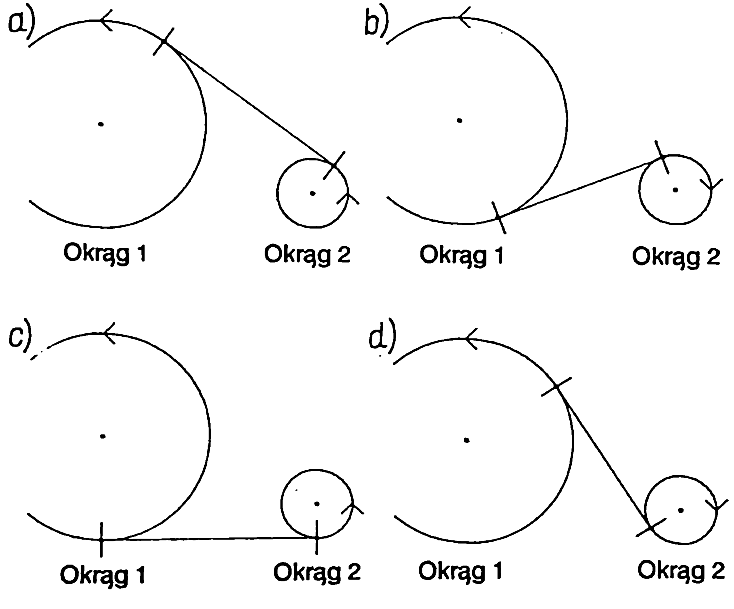
Rys. 8.8. Punkty styczności



Rys. 8.9. Styczne

styczności. Z punktu widzenia grafiki komputerowej ta metoda prowadzi do komplikacji, które najlepiej zilustruje przykład rysowania linii stycznej do dwóch okręgów (rys. 8.9). W ogólnym przypadku można narysować cztery takie linie styczne. Jest potrzebna zatem metoda wyboru jednej ze stycznych, np. metoda określająca kierunek rysowania łuków i prostych.

Na rysunku 8.10a widać, że linia prosta może być uważana za kontynuację Okręgu 2 (narysowanego w kierunku przeciwnym do kierunku ruchu wskazówek zegara) w stronę Okręgu 1 (także narysowanego w kierunku przeciwnym do kierunku ruchu wskazówek zegara). Rozumując podobnie: linia na rys. 8.10b jest kontynuacją Okręgu 1 (narysowanego w kierunku przeciwnym do kierunku ruchu wskazówek zegara) w stronę Okręgu 2 (narysowanego w kierunku zgodnym z kierunkiem ruchu wskazówek zegara); linia na rys. 8.10c jest kontynuacją Okręgu 1 (narysowanego w kierunku przeciwnym do kierunku ruchu wskazówek zegara) w stronę Okręgu 2 (narysowanego w kierunku przeciwnym do kierunku ruchu wskazówek zegara); linia na rys. 8.10d jest kontynuacją Okręgu 2 (narysowanego w kierunku zgodnym z kierunkiem ruchu wskazówek zegara) w stronę Okręgu 1 (narysowanego w kierunku przeciwnym do kierunku ruchu wskazówek zegara). Istnieją inne możliwe kombinacje kierunków, ale po dokładniejszych badaniach dadzą się sprowadzić do pokazanych przypadków.

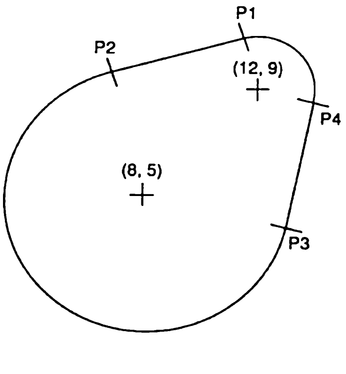


Rys. 8.10. Okręgi skierowane

Do wskazania kierunku kreślenia łuku przyjęto zasadę, że łuk o promieniu dodatnim będzie kreślony w kierunku przeciwnym do kierunku ruchu wskazówek zegara, a o promieniu ujemnym - w kierunku zgodnym z kierunkiem ruchu wskazówek zegara. Kierunek odcinka jest określony przez kolejność współrzędnych jego końców. Podprogram obliczania punktów styczności (przy takim założeniu) pokazano w tabl. 8.10.

Tablica 8.10. Podprogram kreślenia *StyczneDo2Okręgów*

```
Subroutine StyczneDo2Okręgów(X1,Y1,R1,X2,Y2,R2;XP,YP,XQ,YQ)
Real X1,X2,XP,XQ,Y1,Y2,YP,YQ,R1,R2
/* Podprogram oblicza współrzędne punktów styczności (XP,YP) i (XQ,YQ) linii
prostej stycznej do dwóch okręgów, o środkach w punktach (X1,Y1) i (X2,Y2).
Ważna jest kolejność podania okręgów i znak ich promieni. Metoda została
opracowana na Wydziale Informatyki Uniwersytetu w Ljublianie, Jugosławia */
Local Real DX,DY,DL,D1,D2,F,FF,F0,F1,F2,JX,JY
        DX = X2 - X1   /* wektor D od środka do środka */
        DY = Y2 - Y1
        JX = -DY       /* wektor J prostopadły do wektora D */
        JY = DX
        DL = sqrt(DX * DX + DY * DY)   /* długość wektora D */
        F = (R1 - R2)/DL
        FF = F * F
        F0 = sqrt(1 - FF)
        F1 = R1/DL
        F2 = R2/DL
        D1 = F * DX - F0 * JX
        D2 = F * DY - F0 * JY
        XP = X1 + F1 * D1
        YP = Y1 + F1 * D2
        XQ = X2 + F2 * D1
        YQ = Y2 + F2 * D2
End podprogramu StyczneDo2Okregow
```



Rys. 8.11. Figura złożona z okręgów skierowanych i odcinków

Do opisania figury o takim kształcie, jak pokazana na rys. 8.11, należy narysować (w kierunku przeciwnym do kierunku ruchu wskazówek zegara) łuki i odcinki zaczynając od P1, a na P4 kończąc. Można to wykonać za pomocą fragmentu programu pokazanego w tabl. 8.11.

Tablica 8.11. Fragment programu

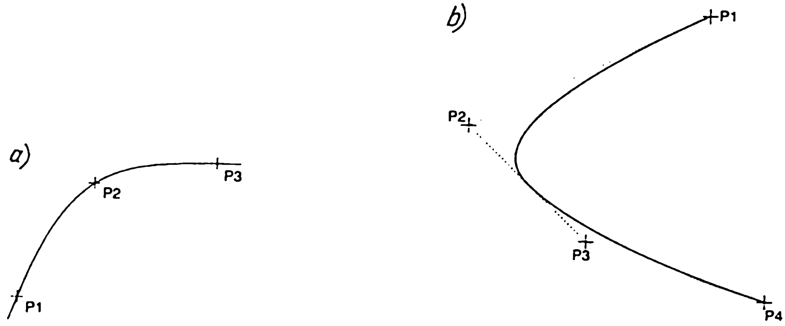
```
/* Fragment programu ładującego do tablicy Out3 współrzędne figury
przedstawionej na rys. 8.11. */
/* Najpierw obliczenie współrzędnych punktów P1 i P2 */
        Call StyczneDo2Okręgów(12,9,2,8,5,5;XP1,YP1,XP2,YP2)
/* Obliczenie współrzędnych punktów P3 i P4 */
        Call StyczneDo2Okręgów(8,5,5,12,9,2;XP3,YP3,XP4,YP4)
/* Obliczenie kątów łuków Łuk1 i Łuk2 */
        Call ObliczKąt(12,9,XP4-12,YP4-9,1;SA1)
        Call ObliczKąt(12,9,XP1-12,YP1-9,1;EA1)
        Call ObliczKąt(8,5,XP2-8,YP2-5,1;SA2)
        Call ObliczKąt(8,5,XP3-8,YP3-5,1;EA2)
/* Opisz Łuk1 i Łuk2 o dodatnich promieniach */
        Call Łuk(72,12,9,SA1,SA1+EA1,2;N1,Out1)
        Call Łuk(72,8,5,SA2,SA2+EA2,5;N2,Out2)
/* Ładuj tablicę Out3 */
        N3 = N1 + N2 + 1
        J = 0
        Dim Out3(N3,2)
        For K = 1 to N1
          J = J + 1
          Out3(J,1) = Out1(K,1)
          Out3(J,2) = Out1(K,2)
        Next K
          For K = 1 to N2
          J = J + 1
          Out3(J,1) = Out2(K,1)
          Out3(J,2) = Out2(K,2)
        Next K
        Out3(N3,1) = Out1(1,1)
        Out3(N3,2) = Out1(1,2)
/* Koniec fragmentu programu */
```

8.5. Krzywe wielomianowe

Krzywe o bardziej skomplikowanych kształtach niż okręgi lub elipsy muszą być opisane w inny sposób. Na przykład za pomocą dyskretyzatora można prześledzić przebieg linii na rysunku, w trybie punktowym. Punkty umieszcza się tak gęsto, jak to jest ko-

nieczne. Śledzenie krzywej nie jest jednak łatwym zadaniem; do rysunku mogą wkraść się różne nierówności, odchyłki i inne zakłócenia. Istnieje wiele technik pozwalających wyeliminować zakłócenia. Jedna z nich polega na wybraniu kilku najważniejszych punktów krzywej i obliczeniu przebiegu na ich podstawie.

Stosuje się dwie metody wyboru punktów. W pierwszej wybiera się punkty, przez które ma przechodzić krzywa (rys. 8.12a). W drugiej wybiera się punkty nie leżące na krzywej, które służą do kontroli jej kształtu (rys. 8.12b). Druga metoda jest popularna w komputerowym projektowaniu, w którym krzywe tworzy się i łączy w procesie konwersacyjnym w celu uzyskania zadowalającego rysunku. Obydwie metody są użyteczne i mają swoje zastosowania.



Rys. 8.12. Punkty; a) na krzywej; b) kontrolne krzywej

Dowolne N punktów można gładko połączyć za pomocą krzywej wielomianowej N – 1 stopnia.

Można zatem wyznaczyć krzywą przechodzącą przez pewną liczbę punktów. Trzy punkty można połączyć krzywą opisaną wielomianem drugiego stopnia:

$$Y = K3 \cdot X^2 + K2 \cdot X + K1$$

Cztery punkty można połączyć krzywą opisaną wielomianem trzeciego stopnia:

$$Y = K4 \cdot X^3 + K3 \cdot X^2 + K2 \cdot X + K1$$

Tak więc, jeśli dane są punkty o współrzędnych: (0,0), (2,5) i (8,6), to stosując zależności algebraiczne można stwierdzić, że leżą one na krzywej (rys. 8.13a) opisanej równaniem:

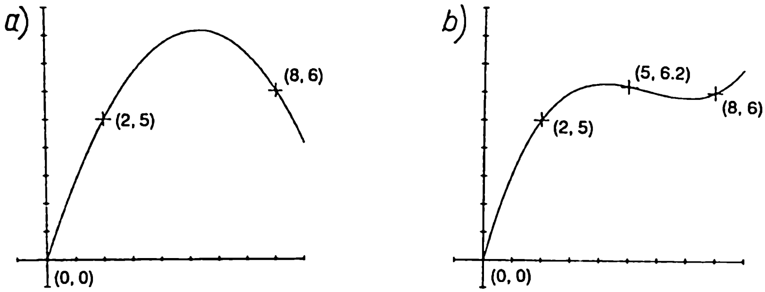
$$Y = -0.29166 \cdot X^2 + 3.0833 \cdot X$$

Współczynniki wielomianu drugiego stopnia wynoszą: $K3 = -0.29166$, $K2 = 3.0833$ oraz $K1 = 0$, ponieważ krzywa przechodzi przez początek układu współrzędnych.

Krzywe otrzymywane z tak prostej aproksymacji nie są tymi, które intuicyjnie dobrze pasują do obrazu. Dla przykładu, wydaje się, że krzywa przedstawiona na rys. 8.13a przechodzi za wysoko między punktami o współrzędnych (2,5) i (8,6). W celu uzyskania dokładnego obrazu krzywej można dodać punkt położony między nimi, np. punkt o współrzędnych (5, 6.2) i aproksymować krzywą za pomocą nowego wielomianu trzeciego stopnia, przechodzącego przez te 4 punkty. Na przykład wielomian

$$Y = 0.42778 \cdot X^3 - 0.719446 \cdot X^2 + 3.76778 \cdot X$$

opisuje krzywą (rys. 8.13b), która z pewnością ma łagodniejszy przebieg niż poprzednia, ale ma lekkie zafalowanie. Zafalowania są charakterystyczne dla krzywych wielomianowych - jest ich tym więcej, im jest wyższy rząd krzywej. Ponadto zauważmy, że w punkcie (8,6) krzywa drugiego stopnia była skierowana ostro do dołu, natomiast krzywa trzeciego stopnia jest skierowana lekko do góry.



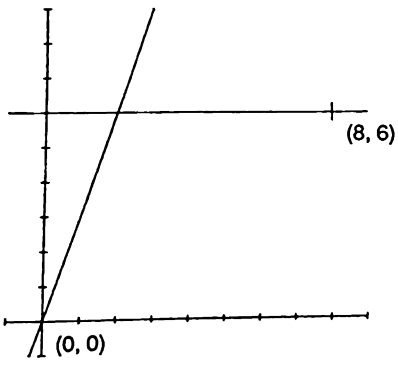
Rys. 8.13. Krzywa wielomianowa; a) drugiego rzędu; b) trzeciego rzędu

Przybliżenie wielomianowe na ogół nie jest dostateczne do uzyskania gładkiego połączenia krzywych z innymi krzywymi lub liniami prostymi. Jeżeli jednak jest znane nachylenie linii, z którymi krzywa ma się łączyć, to można użyć tej informacji w zastosowanej metodzie przybliżania. Mając dane punkty końcowe i nachylenia można dopasować krzywą trzeciego stopnia, a jeśli jest znany jeszcze punkt leżący między nimi, to krzywą czwartego stopnia.

Nachylenie linii prostej obliczamy według zależności:

$$\text{Nachylenie} = (Y2 - Y1) / (X2 - X1).$$

Dla Y2 = Y1 nachylenie wynosi 0, ponieważ linia jest pozioma. Dla X2 = X1 nachylenie jest nieskończone, ponieważ linia jest pionowa. Na rysunku 8.14 przedstawiono linię o nachyleniu równym 3 oraz linię o nachyleniu 0.



Rys. 8.14. Linie proste o różnym nachyleniu

Podprogram PolyFit (tabl. 8.12) pozwala wyznaczyć współrzędne punktów, krzywej przechodzącej przez dwa lub trzy punkty, o zadanym nachyleniu na początku i na końcu. Korzystając z wartości nachylenia i podprogramu PolyFit obliczymy współrzędne krzywej przechodzącej przez punkty (0,0) i (8,6).

Tablica 8.12. Podprogram *PolyFit*

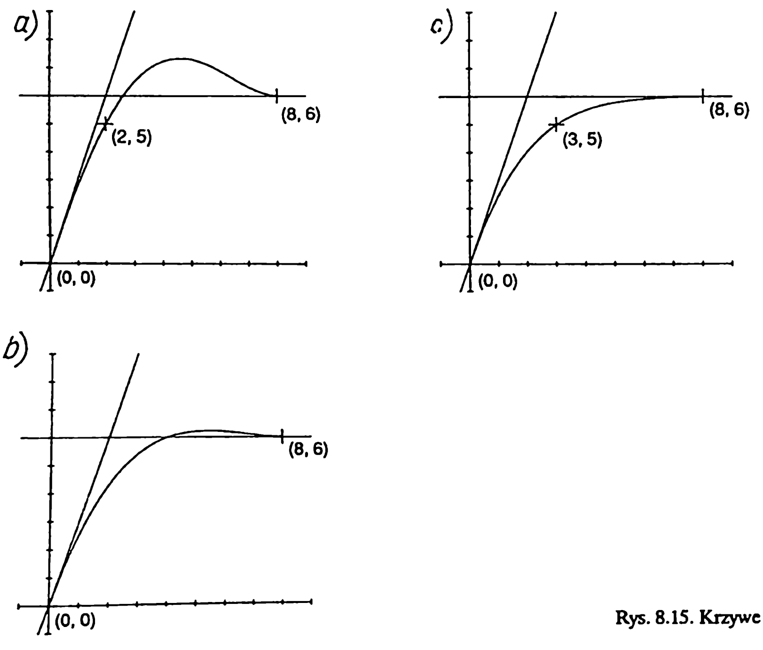
```
Subroutine PolyFit(Num,XS,YS,SS,XE,YE,ES,XM,YM,Inc;N,OA)
Integer Num,N
Real XS,YS,SS,XE,YE,ES,XM,YM,Inc,OA
/* Podprogram ładuje do tablicy OA, N par współrzędnych krzywej wielomianowej
3 stopnia, przechodzącej przez Num = 2 punkty lub krzywej wielomianowej 4 sto-
pnia, przechodzącej przez Num = 3 punkty. SS i ES są nachyleniami linii na po-
czątku i na końcu tej krzywej. Punkt (XM,YM) leży na krzywej. Inc oznacza przy-
rosty wzdłuż osi X. Tablica K zawiera KN współczynników wielomianu. Jeśli są one
potrzebne w innym podprogramie, to powinny być włączone do listy wyników */
Local Integer KN,K9,J
Local Real XX,X1,B1,B2,B3,H,HH,HT,S
        XX = XE - XS
        N = Int(Abs(XX/Inc)) + 1
        KN = 5
        Dim OA(N,2),K(KN)
/* Obliczamy współczynniki */
        K(1) = YS
        K(2) = SS * XX
        If Num = 2 Then
           XM =0
           YM = 0
        Endif
        H = (XM - XS)/XX
        HH = H * H
        B1 = YE - YS - K(2)
        B2 = XX * ES - K(2)
        If Num = 3 Then
           B3 = YM - YS - K(2) * H
           H3 = HH * (HH - 2 * H + 1)
           K(5) = (B3 - B1 * HH - HH * (H - 1) * (B2 - 2 * B1))/H3
        Else
           B3 = YM - YS
           K(5) = 0
           KN = 4
        Endif
        K(4) = B2 - 2 * B1 - 2 * K(5)
        K(3) = YE - YS - K(2) - K(4) - K(5)
```

```
        HT = XS - Inc
/* Obliczamy współrzędne na podstawie współczynników */
        For J = 1 to N
           S = K(1)
           HT = HT + Inc
           X1 = (HT - XS)/XX
           For K9 = 1 to 4
              S = S + K(K9 + 1) * X1 ** K9
           NextK9
           OA(J,1) = HT
           OA(J,2) = 5
        Next J
End podprogramu PolyFit
```

Wielomian czwartego stopnia będzie miał postać:

$$Y = 19.5555 \cdot Xd^4 - 27.111 \cdot Xd^3 - 10.4444 \cdot Xd^2 + 24 \cdot Xd$$

Wyznaczona krzywa przechodzi przez wymienione trzy punkty, ale znów dość wysoko, nim osiągnie płaski przebieg (rys. 8.15a). Xd przyjmuje wartość X/(XS – XE), czyli leży w zakresie od 0 do 1, w przeciwieństwie do poprzedniego przypadku, gdzie X było zmienną rzeczywistą (rzędną).



Rys. 8.15. Krzywe wielomianowe

Wielomian trzeciego stopnia

$$Y = 12 \cdot Xd^3 - 30 \cdot Xd^2 + 24 \cdot Xd$$

opisuje krzywą (rys. 8.15b) o łagodniejszym przebiegu, przechodzącą przez punkty (0,0) i (8,6). Przez zmianę położenia dowolnie ustawionego punktu pośredniego można kontrolować kształt krzywej. Krzywa czwartego stopnia, przechodząca przez punkt (3,5) jest opisana wielomianem:

$$Y = -7.5378 \cdot Xd^4 + 27.0756 \cdot Xd^3 - 37.5378 \cdot Xd^2 + 24 \cdot Xd$$

Opisana krzywa gładko łączy się z liniami o zadanym nachyleniu (rys. 8.15c).
Istnieje wiele innych metod kreślenia krzywych przechodzących przez zadane punkty. Jedna z najbardziej ekonomicznych metod uzyskania krzywej o łagodnym przebiegu, polega na podzieleniu krzywej na krótkie części i aproksymowaniu ich za pomocą wielomianów niskiego stopnia o zadanych nachyleniach na końcach. Należy upewnić się, że nachylenie na końcu jednej części jest takie samo jak nachylenie na początku następnej, oraz skorzystać z punktów pośrednich do kontrolowania kształtu krzywej. Konieczność rysowania obiektów o ładnych kształtach oraz potrzeby grafiki komputerowej i CAD (projektowania wspomaganego komputerem), doprowadziły do techniki, wykorzystującej krzywe parametryczne trzeciego stopnia z punktami kontrolnymi.

8.6. Krzywe parametryczne trzeciego stopnia

Odcinek między punktami (X1,Y1) i (X2,Y2) może być opisany za pomocą jednego parametru *t* i zależności (rozdz. 7)

$$X = X1 + t * (X2 - X1)$$
$$Y = Y1 + t * (Y2 - Y1)$$

w której *t* przyjmuje wartości z zakresu od 0 do 1. Jeśli przyjmiemy, że P1 jest wektorem (X1,Y1), a P2 jest wektorem (X2,Y2), to zależności można sprowadzić do postaci

$$P(t) = P1 + t * (P2 - P1)$$

Jest to wygodny opis skrótowy mający także zastosowanie do opisu linii w przestrzeni trójwymiarowej.

Dla linii przechodzącej między punktami o współrzędnych (X1,Y1,Z1) i (X2,Y2,Z2) jest potrzebna dodatkowa zależność

$$Z = Z1 + t * (Z2 - Z1)$$

Przyglądając się równaniu parametrycznemu, zauważamy, że w miarę przesuwania się wzdłuż prostej (od $t = 0$ do $t = 1$) wzrasta wpływ wektora P2. Dla $t = 0$ P2 nie ma wcale wpływu, dla $t = 1$ odgrywa rolę jedynie P2, a pośrodku wpływ obu wektorów jest jednakowy.

Podobnej techniki można użyć do opisu linii krzywych, ale należy wziąć pod uwagę wpływ większej (niż dwa) liczby punktów. W przypadku linii prostej wpływ punktów końcowych zależy wprost od wartości *t*. Natomiast dla krzywych jest potrzebny bardziej złożony opis, realizowany zwykle za pomocą tzw. *funkcji łączących*. Specjaliści od grafiki komputerowej zaproponowali wiele funkcji łączących i w publikacjach można znaleźć opisy metod ich stosowania.

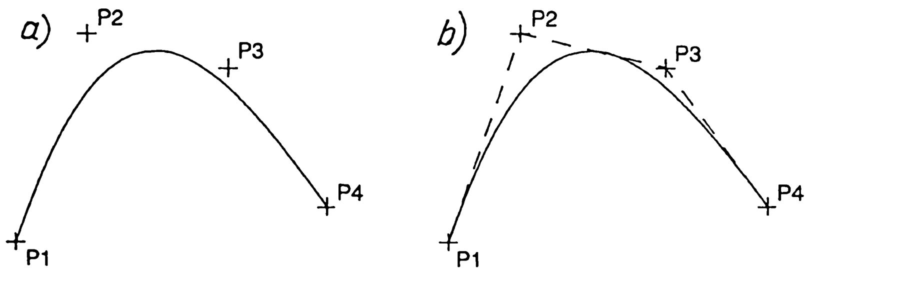
Jedna z najbardziej użytecznych metod do parametrycznego opisu krzywych, została zaproponowana przez H. Timmera z firmy Douglas Aircraft Co., który - podobnie jak inni specjaliści - stosuje cztery punkty (P1, P2, P3, P4) i cztery funkcje łączące (F1, F2, F3, F4). Opis zaproponowany przez H. Timmera ma postać:

$$P(t) = F1(t) \cdot P1 + F2(t) \cdot P2 + F3(t) \cdot P3 + F4(t) \cdot P4$$

gdzie, funkcje łączące są następujące:

$$F1(t) = (1-2 \cdot t) \cdot (1-t)^2$$
$$F2(t) = 4 \cdot t \cdot (1-t)^2$$
$$F3(t) = 4 \cdot t^2 \cdot (1-t)$$
$$F4(t) = (2 \cdot t-1) \cdot t^2$$

Dla wszystkich punktów na krzywej, t przyjmuje wartości z zakresu od 0 do 1. Dla $t = 0$ tylko pierwsza funkcja łącząca jest równa 1, pozostałe są 0, co oznacza, że tylko pierwszy punkt P1 ma wpływ na krzywą. Dla $t = 1$, tylko ostatnia funkcja jest równa 1, pozostałe są 0, co oznacza, że tylko ostatni punkt P4 ma wpływ na krzywą. Dla wartości pośrednich t każda funkcja, a więc każdy punkt ma wpływ na kształt krzywej. Jak pokazano na rys. 8.16a, tylko punkty P1 i P4 leżą na krzywej; pozostałe leżą w pewnej odległości od niej i służą do kontrolowania jej kształtu. Jeśli połączymy te punkty odcinkami prostej (rys. 8.16b), to zobaczymy, że krzywa jest styczna do odcinka P1–P2



Rys. 8.16. Krzywe aproksymujące trzeciego stopnia

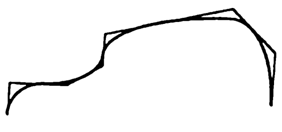
w punkcie P1 i do odcinka P3–P4 w punkcie P4. Dodatkowo, i to jest charakterystyczne dla propozycji Timmera, krzywa dotyka lub przecina w połowie odcinek P2–P3. Jeśli odcinek P1–P4 jest równoległy do odcinka P2–P3, to krzywa jest styczna do odcinka P2–P3 w punkcie wyznaczającym jego połowę. Procedura aproksymacji krzywymi trzeciego stopnia została zrealizowana w postaci podprogramu (tabl. 8.13).

Tablica 8.13. Podprogram *CubicFit*

```
Subroutine CubicFit(Num, Parray; N, Oarray)
Integer Num, N
Real Parray, Oarray
```

```
/* Korzystając z metody Timmera, aproksymujemy parametryczną krzywą 3 stop-
nia, używając 4 punktów o współrzędnych umieszczonych w tablicy Parray o roz-
miarze (4,2). Współrzędne krzywej umieszczamy w tablicy Oarray o rozmiarze
(N,2). Num określa liczbę kroków wynikowej krzywej */
Local Integer Count, K
Local Real A1,A2,F,T,TT,T1,T2
          N = Num + 1
          Dim F(4), Oarray(N,2)
          For Count = 1 to N
             T = (1/Num) * (Count - 1)
             TT = T * T
             T1 = 1 - T
             T2 = T1 * T1
/* Oblicz funkcje łączące */
             F(1) = (1 - 2 * T) * T2
             F(2) = 4 * T * T2
             F(3) = 4 * TT * T1
             F(4) = (2 * T - 1)* TT
             A1 = 0
             A2 = 0
/* Oblicz współrzędne */
             For K = 1 to 4
                A1 = A1 + F(K) * Parray(K,1)
                A2 = A2 + F(K) * Parray(K,2)
             Next K
             Oarray(Count,1) = A1
             Oarray(Count,2) = A2
          Next Count
End podprogramu CubicFit
```

Korzystając z krzywych trzeciego stopnia, z takich jak np. przedstawiona na rys. 8.17, można tworzyć złożone kształty, łącznie z krzywymi zamkniętymi. Ustawienie linii ograniczających powoduje, że poszczególne krzywe gładko przechodzą jedne w drugie, w punktach połączenia. Przy opisie krzywej za pomocą zbioru punktów P1, P2, ...P8 (i ogólnie dowolnej parzystej liczby punktów większej niż cztery) musimy wybrać dodatkowe punkty kontrolne między P3 a P4, między P5 a P6 do wskazania



Rys. 8.17. Aproksymacja za pomocą wielu krzywych trzeciego stopnia

miejsca, w którym krzywe mają się połączyć. Jeśli określono parzystą liczbę N punktów, to trzeba narysować N/2 – 1 segmentów krzywej i tyleż razy wywołać procedurę CubicFit. W pierwszym jej wywołaniu zostaną wzięte pod uwagę punkty P1, P2, P3 i punkt dodatkowy P3/P4; w drugim - punkty P3/P4, P4, P5 i punkt dodatkowy P5/P6; w trzecim - punkty P5/P6, P6, P7 i P8. Łatwo zauważyć, że linie ograniczające dają zgrubne przybliżenie kształtu krzywej.

ZADANIA

8.1. Zakoduj różne, podane w książce, sposoby otrzymywania okręgu. Napisz program testowy do rysowania okręgu o promieniu równym 10, o środku w punkcie (10,10) i porównaj szybkość obliczania każdą metodą. W celu prawidłowego porównania, liczba punktów za każdym razem powinna być taka sama. Zapewne łatwiej będzie określić czas obliczania pięciu okręgów niż jednego, co pozwoli także porównać procedurę OkrągJednostkowy. Pamiętaj, aby wywołać procedurę tylko raz, a następnie pomnożyć otrzymane współrzędne przez 10 i przesunąć okrąg do punktu (10,10), w sposób podany w tabl. 8.7.

8.2. Zakoduj podprogram Łuk3Punkty w swoim komputerze, dodając - we wskazanym miejscu - test na kolinearność. Sprawdź program dla punktów (12.071, 17.071), (13.66, 15) i (10, 1.34). Powinieneś otrzymać okrąg o promieniu prawie równym 10 i o środku w punkcie (5,10). Co się stanie, jeśli dwa punkty wejściowe będą takie same? Umieść w teście sprawdzenie, czy użyte trzy punkty są różne.

8.3. Używając procedury ObliczKąt, potwierdź, że linia przechodząca przez punkty (5,10) i (13.66, 15) tworzy z osią X kąt 30°, a linia między punktami (5,10) i (12.071, 17.071) kąt 135°.

8.4. Korzystając z procedury Łuk3Punkty, przekonaj się, że przez punkty (12.071, 17.071), (13.66, 15) i (10, 1.34) przechodzi łuk o środku w punkcie (5,10), promieniu równym 10, kącie początku 45° i kącie końca równym –60°.

8.5. Zakoduj podprogram PolyFit i przetestuj go na przykładzie podanym w tekście.

8.6. Używając podprogramu CubicFit, narysuj krzywą kontrolowaną przez 4 punkty (0,0), (4,4), (6,5) i (8,0). Powtórz ćwiczenie dla punktów (0,0), (4,–4), (6,5) i (8,0). Przekonaj się, że linie ograniczające wyznaczają w zarysie przebieg krzywej.

8.7. Napisz program korzystający z więcej niż 4 punktów kontrolnych do kreślenia złożonych krzywych. Program ten powinien korzystać z podprogramu CubicFit tyle razy, ile będzie to potrzebne.

9. Przekształcenia na płaszczyźnie

Po opracowaniu modelu dwuwymiarowego obiektu, który będzie wyświetlany, można zająć się jego umieszczeniem i orientacją w przestrzeni współrzędnych świata. W celu uzyskania pożądanego rysunku, trzeba obiekt przesunąć, obrócić lub zmienić jego wymiary. Również ze względu na zmniejszenie nakładu pracy projektanta, jest wygodnie opisać obiekt w określonej pozycji i orientacji. Na przykład, znacznie łatwiej jest określić współrzędne kwadratu jednostkowego o środku w początku układu współrzędnych i bokach równoległych do osi, niż kwadratu o bokach 13.46×13.46 obróconego o 42° i środku w punkcie (15.4, 36.276).

Właściwe przesunięcie, pozycji i orientacji, uzyskuje się za pomocą pewnych przekształceń geometrycznych lub *transformacji* modelu. Rozróżniamy następujące transformacje:

- przesunięcie, czyli przemieszczenie biektu do właściwej pozycji,
- obrót w płaszczyźnie, czyli nadanie obiektowi właściwej orientacji,
- skalowanie, czyli zmiana wymiarów i proporcji obiektu,
- odkszatłcenie, czyli zmiana kształtu obiektu na skutek różnego przesunięcia niektórych jego wierzchołków.

Przesunięcie, obrót i skalowanie są typowymi transformacjami i każdy system graficzny powinien łatwo je realizować. Natomiast odkształcenie jest rzadko stosowane.

9. 1. Przesunięcie

Dokonując przesunięcia obiektu dwuwymiarowego, dodajemy wartość przesunięcia w kierunku X do wszystkich współrzędnych X modelu, a wartość przesunięcia w kierunku Y do wszystkich współrzędnych Y. Dla danego przesunięcia Xmv w kierunku X i danego przesunięcia Ymv w kierunku Y można obliczyć nowe (Xn,Yn) współrzędne na podstawie poprzednich (Xp,Yp), za pomocą równań

$$Xn = Xp + Xmv$$
$$Yn = Yp + Ymv$$

Aby przesunąć całą figurę wystarczy przesunąć końce jej boków do nowych pozycji; Xmv jest ujemne przy przesuwaniu w lewo, a Ymv jest ujemne przy przesuwaniu w dół.

9.2. Obrót na płaszczyźnie

Jeżeli punkt o współrzędnych (X,Y) jest obracany wokół początku układu współrzędnych o A^{o}, to współrzędne nowego położenia (Xn,Yn) są określane następująco

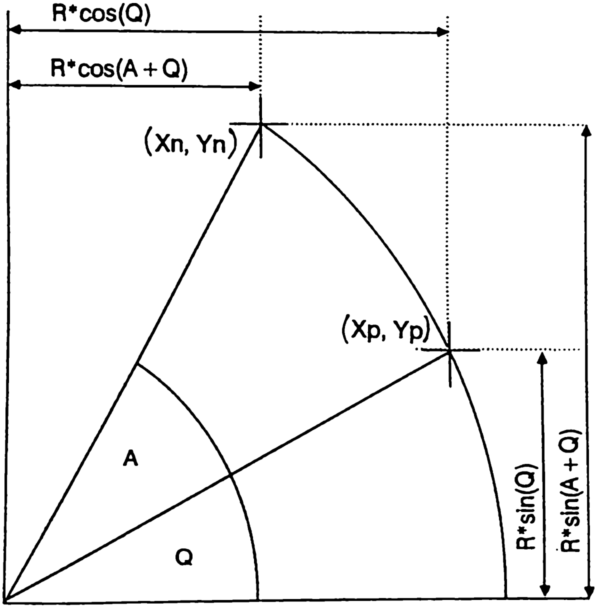
$$Xn = Xp * \cos(A) - Yp * \sin(A)$$
$$Yn = Xp * \sin(A) + Yp * \cos(A)$$

Dla dodatnich wartości kąta A obrót następuje w kierunku przeciwnym do ruchu wskazówek zegara; natomiast dla ujemnych - w kierunku zgodnym z ruchem wskazówek zegara.

W notacji macierzowej zależności te można zapisać w postaci:

$$(Xn,Yn) = (Xp,Yp) * \begin{bmatrix} \cos(A) & \sin(A) \\ -\sin(A) & \cos(A) \end{bmatrix}$$

Zapis macierzowy i operacje na macierzach są omówione w Dodatku 1. Do zademonstrowania poprawności tej formuły skorzystamy, raz jeszcze, ze wzoru na cosinus sumy kątów (rys. 9.1).



Rys. 9.1. Obrót punktu na płaszczyźnie

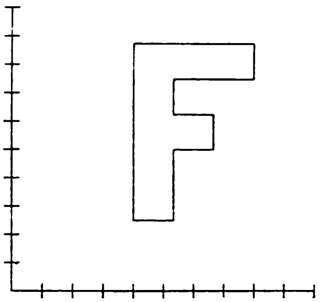
```
Xn = R * cos(A + Q) =
        = R * (cos(A) * cos(Q) - sin(A) * sin(Q)) =
        = R * cos(Q) * cos(A) - R * sin(Q) * sin(A) =
        = Xp * cos(A) - Yp * sin(A)

Yn = R * sin(A + Q) =
        = R * (sin(A) * cos(Q) + sin(Q) * cos(A)) =
        = R * cos(Q) * sin(A) + R * sin(Q) * cos(A) =
        = Xp * sin(A) + Yp * cos(A)
```

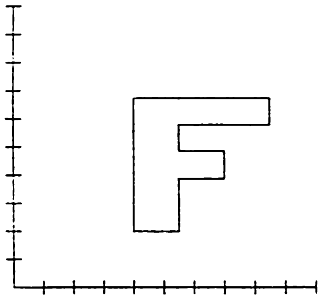
Wzory te opisują metodę obrotu obiektu na płaszczyźnie XY. Oczywiście jest możliwe także dokonanie obrotu obiektu płaskiego w przestrzeni trójwymiarowej; zostanie to przedstawione przy omawianiu obrotów w przestrzeni trójwymiarowej.

9.3. Skalowanie

Jeżeli dokonujemy równomiernego skalowania obiektu w obu kierunkach X i Y, to zmieniają się wymiary obiektu (i czasami jego położenie względem początku układu współrzędnych), ale jego proporcje nie ulegają zmianie (rys. 9.2). Jest możliwe także zastosowanie różnych skal w kierunkach X i Y, co powoduje zmianę wymiarów, proporcji i czasami położenia (rys. 9.3).



Rys. 9.2. Skalowanie litery F w stosunku 1.25



Rys. 9.3. Skalowanie litery F tylko w kierunku X

Skalowanie wykonuje się mnożąc każdą ze współrzędnych przez odpowiedni współczynnik skalowania. Tak więc

```
Xn = Xp * Xsk
Yn = Yp * Ysk
```

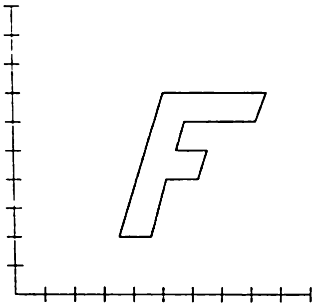
lub w zapisie macierzowym

$$(Xn,Yn) = (Xp,Yp) * \begin{bmatrix} Xsk & 0 \\ 0 & Ysk \end{bmatrix}$$

Przy współczynniku skalowania większym niż 1, figura jest rozciągana; współczynnik o wartości z zakresu od 0 do 1 powoduje zmniejszanie figury. Ujemny współczynnik skalowania powoduje powstanie różnych odbić. Zbadanie wyników takiego skalowania pozostawiamy Czytelnikowi jako ćwiczenie.

9.4. Odkształcanie

Zastosowanie transformacji odkształcania, powoduje przesuwanie różnych elementów obiektu o różne wartości, co powoduje zniekształcenie obiektu (rys. 9.4).



Rys. 9.4. Odkształcenie litery F

Odkształcenie w kierunku Y jest określone zależnościami

$$Xn = Xp$$
$$Yn = Yp + Xp * Yod$$

Odkształcenie w kierunku X jest określone zależnościami

$$Xn = Xp + Yp * Xod$$
$$Yn = Yp$$

Zapis macierzowy ma postać:

$$(Xn,Yn) = (Xp,Yp) * \begin{bmatrix} 1 & Yod \\ 0 & 1 \end{bmatrix}$$

$$(Xn,Yn) = (Xp,Yp) * \begin{bmatrix} 1 & 0 \\ Xod & 1 \end{bmatrix}$$

9.5. Współrzędne jednorodne

Podane macierzowe metody realizacji poszczególnych transformacji nie obejmowały przesunięcia. Nie jest to przeoczenie; wynika to z natury układu współrzędnych, który

był stosowany dotychczas. W normalnym układzie współrzędnych, mnożenie macierzy nie pozwala na realizację przesunięcia. Jest jednak inna postać współrzędnych, używana powszechnie w grafice komputerowej, która przedstawia operacje na macierzach w sposób bardziej konsekwentny i być może łatwiejszy do zrozumienia. Jest to tzw. *system współrzędnych jednorodnych.*

Współrzędne jednorodne zostały pierwotnie opracowane w celu ułatwienia matematykom dowodzenia twierdzeń z zakresu geometrii rzutowej - działu matematyki zajmującego się m.in. perspektywą. W grafice komputerowej, współrzędne jednorodne są używane głównie ze względu na możliwość połączenia oddzielnych macierzy przesunięcia, obrotu, skalowania i (jak zobaczymy później) perspektywy do postaci jednej macierzy, która może być używana do transformowania współrzędnych za pomocą mnożenia.

Wymiar współrzędnych jednorodnych jest o jeden większy od wymiaru opisywanej przestrzeni. I tak, współrzędne jednorodne dla przestrzeni dwuwymiarowej mają postać (Wx,Wy,W), a dla przestrzeni trójwymiarowej postać (Wx,Wy,Wz,W). Współrzędne jednorodne mogą być przywrócone do pierwotnego wymiaru w wyniku podzielenia ich przez współrzędną W (jest ona zawsze różna od zera) i pominięcie jej. Stąd (Wx,Wy,W) zamienia się w (X,Y), a (Wx,Wy,Wz,W) przyjmuje postać (X,Y,Z).

Ponieważ W może przyjąć dowolną wartość różną od zera, to punkt we współrzędnych jednorodnych może mieć nieskończenie wiele reprezentacji. Na przykład, punkt (2,3) w przestrzeni dwuwymiarowej może być reprezentowany w systemie współrzędnych jednorodnych jako: (2,3,1), gdzie W = 1, (1,1.5,0.5), gdzie W = 0.5, lub (18,27,9), gdzie W = 9 itd. Współrzędna W może być traktowana jako współczynnik skalowania, który dla większości zastosowań ma wartość 1.

Przesunięcie

Przesunięcie punktu (Xp,Yp) o wartości Xmv, Ymv jest opisane macierzą

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Xmv & Ymv & 1 \end{bmatrix}$$

Mnożąc wektor (Xp,Yp,1) przez tę macierz, otrzymujemy nowe współrzędne

$$(Xn,Yn,1) = (Xp,Yp,1) * \mathbf{T} = (Xp + Xmv,\ Yp + Ymv,\ 1)$$

Zauważmy, że korzystając ze współrzędnych jednorodnych można operację przesunięcia (która wymaga tylko dodawania) zrealizować za pomocą mnożenia.

Obrót

Obrót o kąt A opisujemy za pomocą macierzy

$$\mathbf{R} = \begin{bmatrix} C & S & 0 \\ -S & C & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

gdzie C = cos(A) i S = sin(A).

Mnożąc wektor (Xp, Yp, 1) przez macierz **R**, otrzymujemy nowe współrzędne

$$(Xn,Yn,1) = (Xp,Yp,1) * \mathbf{R} = (Xp * C - Yp * S, Xp * S + Yp * C, 1).$$

Skalowanie

Skalowanie położenia punktu o wielkość Xsk, Ysk jest opisane macierzą

$$S = \begin{bmatrix} Xsk & 0 & 0 \\ 0 & Ysk & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Mnożąc wektor (Xp, Yp, 1) przez macierz S, otrzymujemy nowe współrzędne

$$(Xn,Yn,1) = (Xp,Yp,1) * S = (Xp * Xsk, Yp * Ysk, 1)$$

9.6. Operacje odwrotne

Istnieje możliwość (bardzo istotna) wykonywania operacji odwrotnych, tzn. odwracających skutek transformacji macierzowych. Odwrotne operacje do podanych transformacji realizuje się następująco:

- dla macierzy przesunięcia należy zostosować przesunięcie –Xmv, –Ymv, 1;
- dla macierzy obrotu należy zostosować ujemny kąt obrotu;
- dla macierzy skalowania należy zastosować 1/Xsk, 1/Ysk, 1.

To proste podejście zawodzi w odniesieniu do macierzy, która została otrzymana w wyniku pomnożenia macierzy obrotu, skalowania i przesunięcia - procesu określanego jako łączenie (*konkatenacja*). W tym przypadku w celu uzyskania efektu odwrotnego należy zastosować standardowe metody algebry macierzy.

9.7. Łączenie

Do wykonania złożonych ruchów i obrotów można korzystać z kolejnych, oddzielnych macierzy lub z jednej macierzy utworzonej w wyniku łączenia transformacji. Druga metoda jest bardziej efektywna wówczas, gdy trzeba dokonać transformacji dużej liczby punktów. Obrót w płaszczyźnie XY wokół punktu (Xpt, Ypt, 1) jest opisany zależnością:

$$(Xn,Yn,1) = (Xp,Yp,1) * \mathbf{T} * \mathbf{R} * \mathbf{inv(T)}$$

gdzie

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -Xpt & -Ypt & 1 \end{bmatrix}$$

$$R = \begin{bmatrix} C & S & 0 \\ -S & C & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$inv(T) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ Xpt & Ypt & 1 \end{bmatrix}$$

Dokonuje się więc następujących operacji: przesunięcia obiektu do początku układu współrzędnych, obrótu wokół początku układu współrzędnych, przesunięcia odwrotnego do wykonanego na początku. W wyniku uzyskuje się obrót obiektu na płaszczyźnie wokół punktu (Xpt, Ypt).

W programie zazwyczaj tworzy się nową macierz Con określoną następująco:

Con = T * R * inv(T)

i korzysta się z niej zamiast z trzech niezależnych macierzy.

Jako ćwiczenie pozostawiamy wykazanie, że

$$Con = \begin{bmatrix} C & S & 0 \\ -S & C & 0 \\ A & B & 1 \end{bmatrix}$$

przy czym

A = –Xpt * C + Ypt * S + Xpt
B = –Xpt * S – Ypt * C + Ypt

Trzeba podkreślić, że - ze względu na szybkość obliczeń - korzystanie z gotowej formuły jest lepsze niż korzystanie z wersji macierzowej tej formuły. Wersję tę można stosować w komputerowym systemie przystosowanym do efektywnego działania na macierzach, lub wówczas, gdy operacje na macierzach są elementem języka, jak np. w języku APL albo w niektórych wersjach Basicu. W większości podręczników i publikacji z dziedziny grafiki komputerowej wykorzystuje się jednak macierzowy opis transformacji, dlatego też jest wskazane zaznajomienie się z tymi metodami.

ZADANIA

9.1. Napisz i przetestuj program realizujący przesunięcie w stosunku do zbioru N współrzędnych. Wejściowymi parametrami programu powinno być N, nazwy tablic zawierających współrzędne X i Y oraz wielkości przesunięcia Xmv i Ymv. W wyniku, program powinien dawać liczbę par punktów Num i tyleż par punktów X,Y w tablicy wyjściowej.

9.2. Korzystając z programu pokaż, że odcinek o końcach w punktach (5,7) i (9,9) po przesunięciu o –6 jednostek w kierunku X i 3,3 w kierunku Y, stanie się odcinkiem o końcach w punktach (–1,10.3) i (3,12.3).

9.3. Napisz i sprawdź program obrotu zbioru podobnych punktów. Kąt obrotu powinien być podawany w stopniach (jeśli system wymaga określania kąta w radianach, to program powinien zapewnić odpowiednią konwersję). Należy

pamiętać o tym, że programy transformacji są wielokrotnie wywoływane przez programy grafiki dwuwymiarowej i dlatego powinny być napisane możliwie starannie tak, żeby zapewnić ich dużą efektywność. W szczególności trzeba korzystać z dostępnej w systemie największej dokładności. Nie należy dokonywać zaokrąglania liczb całkowitych, jeżeli nie zamierza się tych punktów wyświetlać lub wykreślać.

9.4. Korzystając z programu sprawdź, że odcinek o końcach w punktach (5,7) i (9,9) po obrocie o 35° wokół początku układu współrzędnych staje się odcinkiem o końcach w punktach (0.081,8.6) i (2.21,12.535). Dokonaj obrotu pierwotnego odcinka o -40°. Wykaż, że jego końce mają współrzędne (8.33,2.148) i (12.68,1.11).

9.5. Dokonaliśmy obrotu odcinka wokół początku układu współrzędnych. W tym zadaniu dokonujemy obrotu odcinka wokół punktu końcowego (5,7). W tym celu należy przesunąć ten punkt końcowy do początku układu współrzędnych, dokonać obrotu i przesunąć odcinek tak, żeby punkt końcowy stanowiący środek obrotu wrócił do pierwotnego położenia. Oznacza to konieczność dwukrotnego wywołania programu przesuwania. Współrzędne końców odcinka obróconego o 35° wokół punktu (5,7) powinny mieć wartość (5,7) i (7.13,10.93); przy obrocie o -40° odpowiednio (5,7) i (9.35,5.96).

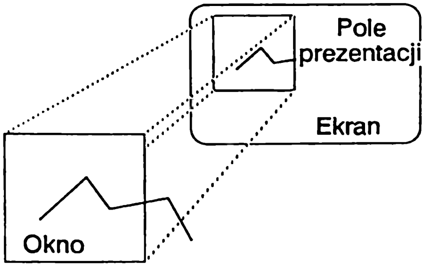
9.6. Korzystając ze współrzędnych jednorodnych tak napisz program z zadania 9.5, żeby można było wykonywać tylko mnożenie macierzy. Wykorzystaj metodę podaną w punkcie dotyczącym łączenia macierzy.

10. Obcinanie dwuwymiarowe i odwzorowanie okna na pole prezentacji

Następnym zadaniem w procesie otrzymywania obrazu graficznego jest jego obcięcie do wymiarów powierzchni, na której ten obraz będzie wyświetlany. Obcięcie polega na wyeliminowaniu wszystkich tych fragmentów linii, które nie mieszczą się w oknie, a następnie na odwzorowaniu okna na pole prezentacji (odpowiednie pole na ekranie). Niektóre systemy graficzne pozwalają na bezpośrednie wykonywanie takich zadań, dzięki możliwości korzystania z poleceń definiujących okna i pole prezentacji. Polecenia mogą mieć postać:

Obetnij (Tak);
Okno (Ol,Op,Od,Og);
Pole prezentacji (Pl,Pp,Pd,Pg).

Polecenia odwzorowują okno na pole prezentacji, w sposób pokazany na rys. 10.1 i zapewniają wyświetlanie tylko tych punktów, które leżą wewnątrz pola prezentacji. W wielu systemach, zwłaszcza w komputerach osobistych, takie udogodnienia jak obcinanie i odwzorowanie okno/pole prezentacji nie są dostępne. Należy zatem napisać swoje własne programy, po to, żeby uchronić się przed niepożądanymi efektami, które mogłyby wystąpić w przypadku rysowania poza dostępnym obszarem. Jeśli nie podejmiemy odpowiednich działań, mogą wystąpić trzy rodzaje zjawisk:



Rys. 10.1. Odwzorowanie okna na pole prezentacji i obcinanie

- linie będą się "zawijały" wychodząc z jednego boku ekranu i wracając z przeciwnej strony, zakłócając tym samym obraz (dodatkowe odcinki);
- w przypadku ploterów, piórko może uderzać o krawędź powierzchni kreślenia; stwarza to potencjalną możliwość uszkodzenia urządzenia;
- system może się po prostu zawiesić, uniemożliwiając tym samym jakąkolwiek dalszą pracę, do czasu restartu.

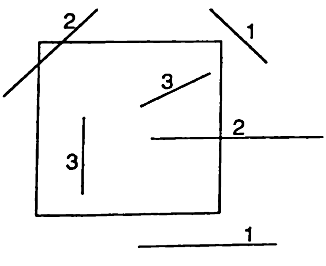
Operacje wykorzystywane przy odwzorowaniu i obcinaniu na płaszczyźnie są operacjami prostymi. W celu uzyskania odpowiedniej szybkości obcinania, programy realizujące te algorytmy powinny być napisane w kodzie maszynowym.

10.1. Obcinanie

Konieczność obcinania pojawia się wtedy, kiedy jeden (lub oba) koniec jakiegoś odcinka leży poza oknem i w związku z tym nie powinien być widoczny. Trzeba rozważyć trzy przypadki:
- cały odcinek jest niewidoczny,
- tylko część odcinka jest widoczna,
- cały odcinek jest widoczny.

Jeżeli oba końce odcinka leżą wewnątrz okna, to cały odcinek jest widoczny. Jeżeli jeden koniec odcinka jest w oknie, a drugi nie, to tylko część odcinka jest widoczna. Są to proste sytuacje do rozwiązania. Większy kłopot występuje wówczas, gdy oba końce odcinka leżą na zewnątrz okna. W takim przypadku albo cały odcinek jest niewidoczny, albo widoczna jest tylko jego część środkowa. Możliwości te są pokazane na rys. 10.2. Każdy algorytm obcinania musi uwzględniać wymienione przypadki i powinien zawierać sprawdzenie, który przypadek występuje:
- odcinek całkowicie niewidoczny,
- odcinek częściowo widoczny,
- odcinek w całości widoczny.

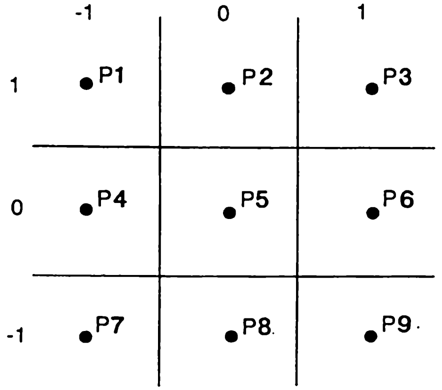
Następnie, dla:
- przypadku 1 algorytm powinien pominąć odcinek,
- przypadku 2 znaleźć przecięcie odcinka z brzegami okna i dołączyć nowy odcinek,
- przypadku 3 uwzględnić cały odcinek.



Rys. 10.2. Obcinanie odcinka. Odcinki typu 1 są niewidoczne. Odcinki typu 3 są w całości widoczne. Odcinki typu 2 są częściowo widoczne i trzeba wyznaczyć ich przecięcia z brzegami okna

W przypadku odcinka częściowo widocznego musimy mieć możliwość obliczenia punktu przecięcia dwóch odcinków: odcinka rozpatrywanego i jednego z brzegów okna. Brzegi są odcinkami: poziomym i pionowym, co upraszcza obliczenia. Na początku jednak musimy sprawdzić, z którym z trzech przypadków mamy do czynienia. W tym celu jest potrzebny program, który dokona klasyfikacji odcinka na podstawie analizy położenia jego końców. Jeden z takich programów wykorzystuje algorytm Cohena-Sutherlanda. Twórcy tego algorytmu byli pionierami grafiki komputerowej.

W oryginalnej postaci algorytm wykorzystuje pomysłową metodę działania na bitach, która jest wygodna wówczas, gdy programy są pisane w kodzie maszynowym. W celu uogólnienia, w książce przedstawiono zmodyfikowaną postać algorytmu, która upraszcza programowanie na wyższym poziomie. Przy założeniu, że okno leży w środku siatki 3×3, której poszczególne pola mają przypisane kody jak pokazano na rys. 10.3, każdemu punktowi w siatce, może być jednoznacznie przypisana, zależnie od



Rys. 10.3. Kody obcinania

położenia, para wartości Xkod, Ykod. W tablicy 10.1 przedstawiono możliwe kombinacje kodów obcinania. Zauważmy, że dla odcinka, którego końce znajdują się wewnątrz środkowego kwadratu (a więc odcinka całkowicie widocznego) zarówno Xkod jak i Ykod są równe zeru. Za pomocą dwóch programów: KodyObcinania (tabl. 10.2) i ObcinanieOdcinka (tabl. 10.3) można dokonać przypisania kodów i obcięcia odcinka.

Tablica 10.1. Kody obcinania

Punkt	Xkod	Ykod
P1	-1	1
P2	0	1
P3	1	1
P4	-1	0
P5	0	0
P6	1	0
P7	-1	-1
P8	0	-1
P9	1	-1

Tablica 10.2. Podprogram *KodyObcinania*

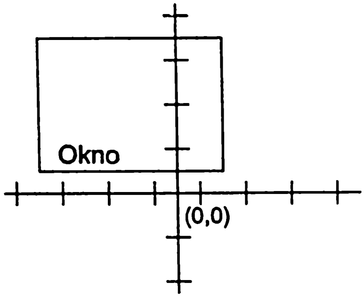
```
Subroutine KodyObcinania(Xw,Yw;Xkod,Ykod)
Integer Xkod,Ykod
Real Xw, Yw
/* Podprogram oblicza Xkod i Ykod dla dowolnego punktu (Xw, Yw) i dla globalnie
zdefiniowanego okna. Środek okna jest przesunięty o Xprz i Yprz w stosunku do
środka układu współrzędnych*/
Global Real Xprz, Yprz, Xbrz, Ybrz
Lokal Real Xn, Yn
        Xn = Xw - Xprz
        Yn = Yw - Yprz
        If Abs(Xn) > Xbrz Then
           Xkod = Sgn(Xn)
        Else
           Xkod = 0
        Endif
        If Abs(Yn) > Ybrz Then
           Ykod = Sgn(Yn)
        Else
           Ykod = 0
        Endif
End podprogramu KodyObcinania
```

Tablica 10.3. Podprogram *ObcinanieOdcinka*

```
Subroutine ObcinanieOdcinka(X1p,Y1p,X2p,Y2p; X1n,Y1n,X2n,Y2n,Wid)
Integer Wid
Real X1p, Y1p, X2p, Y2p, X1n, Y1n
/* Podprogram określa punkty (X1n, Y1n) i (X2n, Y2n) po obcięciu dla zadanych
punktów (X1p, Y1p) i (X2p, Y2p) */
Local Real Stosunek
/* Znajdź kody obcinania dla punktów końcowych */
        Call KodyObcinania(X1p,Y1p;X1kod,Y1kod)
        Call KodyObcinania(X2p,Y2p;X2kod,Y2kod)
/* Pomiń punkty, które leżą w samym obszarze poza ekranem */
        If X1kod * X2kod = 1 or Y1kod * Y2kod = 1 Then
           Wid = 0   /* Odcinek niewidoczny */
           X1n = X1p
           Y1n = Y1p
           X2n = X2p
           Y2n = Y2p
        Else
/* Przesuń odcinek do okna umieszczonego w środku */
           X1n = X1p - Xprz
```

```
            Y1n = Y1p - Yprz
            X2n = X2p - Xprz
            Y2n = Y2p - Yprz
/* Obetnij odcinek od punktu 1 do najbliższego brzegu X */
        If X1kod < > 0 Then
            Xpom = Xbrz * X1kod
            Stosunek = (Xpom - X1n)/(X2n - X1n)
            Y1n = Y1n + (Y2n - Y1n) * Stosunek
            X1n = Xpom
/* Znajdź kody obcinania dla skróconego odcinka */
            Call KodyObcinania(X1n + Xprz,Y1n + Yprz; X1kod,Y1kod)
        Endif
/* Obetnij odcinek od punktu 1 do najbliższego brzegu Y */
        If Y1kod < > 0 Then
            Ypom = Ybrz * Y1kod
            Stosunek = (Ypom - Y1n)/(Y2n - Y1n)
            X1n = X1n + (X2n - X1n) * Stosunek
            Y1n = Ypom
        Endif
/* Obetnij odcinek od punktu 2 do najbliższego brzegu X */
        If X2kod < > 0 Then
            Xpom = Xbrz * X2kod
            Stosunek = (Xpom - X1n)/(X2n - X1n)
            Y2n = Y1n + (Y2n - Y1n) * Stosunek
            X2n = Xpom
/* Znajdź kody obcinania dla skróconego odcinka */
            Call KodyObcinania(X2n + Xprz,Y2n + Yprz; X2kod,Y2kod)
        Endif
/* Obetnij odcinek od punktu 2 do najbliższego brzegu Y */
        If Y2kod < > 0 Then
            Ypom = Ybrz * Y2kod
            Stosunek = (Ypom - Y1n)/(Y2n - Y1n)
            X2n = X1n + (X2n - X1n) * Stosunek
            Y2n = Ypom
        Endif
        Wid = 1   /* Pewna część jest widoczna */
/* Przesuń odcinek z powrotem do punktu początkowego */
        X1n = X1n + Xprz
        Y1n = Y1n + Yprz
        X2n = X2n + Xprz
        Y2n = Y2n + Yprz
        Endif
End podprogramu ObcinanieOdcinka
```

Użyte oznaczenia: p - wartość poprzednia, n - wartość nowa, prz - wartość przesunięcia, brz - wartość brzegowa, pom - wartość pomocnicza.



Rys. 10.4. Okno przesunięte w stosunku do początku układu współrzędnych

Programy te są poprawne przy założeniu, że środek okna pokrywa się z początkiem układu współrzędnych. Jednak nie zawsze tak jest, (np. jak w przypadku okna przedstawionego na rys. 10.4). Należy więc wyznaczyć wartość przesunięcia okna i odcinków za pomocą programu Ustaw-okno przedstawionego w tabl. 10.4.

Tablica 10.4. Podprogram *Ustaw-okno*

```
Subroutine Ustaw-okno(Wxmin,Wxmax,Wymin,Wymax)
Real Wxmin,Wxmax,Wymin,Wymax
/* Podprogram ustawia globalne parametry okna */
Global Real Wl,Wp,Wd,Wg,Xprz,Yprz,Xbrz,Ybrz
      Wl = Wxmin
      Wp = Wxmax
      Wd = Wymin
      Wg = Wymax
/* Sprawdź, czy parametry są we właściwym porządku */
      If Wl > = Wp or Wd > = Wg Then
         Print 'Definicja okna jest niepoprawna'
         Stop
      Endif
/* Oblicz wartość przesunięcia środka okna względem początku układu
współrzędnych */
      Xprz = (Wp + Wl) * 0.5
      Yprz = (Wg + Wd) * 0.5
/* Określ nowe brzegi */
      Xbrz = Wp – Xprz
      Ybrz = Wg – Xprz
End podprogramu Ustaw-okno
```

Użyte oznaczenia: l - lewo, p - prawo, d - dół, g - góra, brz - brzeg.

Wymiary pola prezentacji są określone przez program Ustaw-wsPolePrezentacji. Zakłada się, że wcześniej zostały podane w programie minimalne i maksymalne wymiary Xvx, Yvx, Xvn, Yvn dla urządzenia. Wymiary mają wartości całkowite. Program Ustaw-wsPolePrezentacji jest przedstawiony w tabl. 10.5.

Tablica 10.5. Podprogram *Ustaw-wsPolePrezentacji*

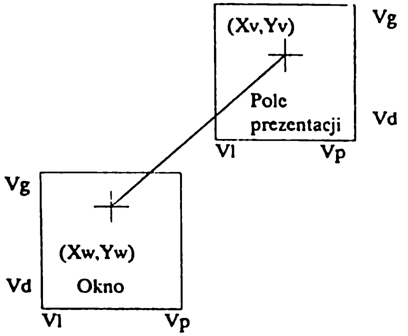
```
Subroutine Ustaw-wsPolePrezentacji(Vxmin,Vxmax,Vymin,Vymax)
Integer Vxmin, Vxmax, Vymin, Vymax
/* Program określa globalne wymiary pola prezentacji we współrzędnych
urządzenia*/
Global Integer Vl, Vp, Vd, Vg, Xvx, Yvx, Xvn,Yvn
        Vl = Vxmin
        Vp = Vxmax,
        Vd = Vymin
        Vg = Vymax
/* Sprawdź czy parametry są we właściwym porządku */
        If Vl > = Vp or Vd > = Vg Then
           Print 'Definicja pola prezentacji jest niepoprawna'
           Stop
        Endif
/* Sprawdź, czy żądane pole prezentacji nie jest większe od dopuszczalnego */
        If Vl < Xvn or Vp > Xvx or Vd < Yvn or Vg > Yvx Then
           Print 'Wymiary pola prezentacji są niepoprawne'
           Stop
        Endif
End podprogramu Ustaw-wsPolePrezentacji
```

10.2. Odwzorowanie okna na pole prezentacji

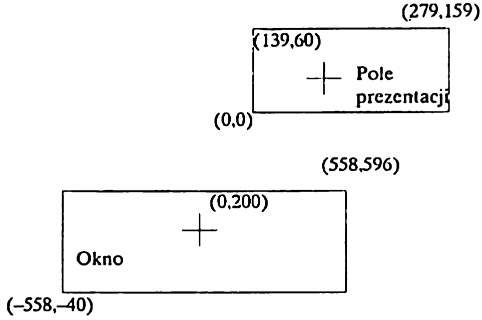
Do obliczeń związanych, z odwzorowaniem okno-pole prezentacji ze współrzędnych świata na współrzędne urządzenia dla dowolnego punktu (Xw,Yw), korzystamy z oznaczeń takich jak przedstawiono na rys. 10.5 i następujących zależności:

$$Xsk = (Vp - Vl)/(Wp - Wl)$$
$$Ysk = (Vg - Vd)/(Wg - Wd)$$
$$Xv = (Xw - Wl) * Xsk + Vl$$
$$Yv = (Yw - Wd) * Ysk + Vd$$

Powyższe zależności przypisują punktowi (Xw,Yw) w układzie współrzędnych świata, punkt (Xv,Yv) w układzie współrzędnych urządzenia. Wartości Xsk i Ysk należy obliczyć tylko raz, bezpośrednio po określeniu wymiarów okna i pola prezentacji. Czytelnikowi proponujemy napisanie programu o nazwie WyznaczSkale, który będzie wykonywał te obliczenia. Obliczenie wartości (Xv,Yv) odpowiadających (Xw,Yw) trzeba wykonywać dla punktów końcowych wszystkich odcinków, które mają być wyświetlone. Można w tym celu wykorzystać program Zmianawspółrz.



Rys. 10.5. Okno i pole prezentacji



Rys. 10.6. Odwzorowanie okna na pole prezentacji

Czasami chcemy, żeby pole prezentacji pokrywało całą powierzchnię ekranu, kiedy indziej jest potrzebna tylko część tej powierzchni. Dla urządzenia o rozdzielczości 280 × 160, oknie o bokach od –558 do 558 i od –40 do 596, oraz polu prezentacji pokrywającym całą powierzchnię ekranu (rys. 10.6), stosując te zależności otrzymamy następujące dane:

Vp = 279 Wp = 558
Vl = 0 Wl = –558
Vg = 159 Wg = 596
Vd = 0 Wd = –40
Xsk = 279/1116 = 0.25
Ysk = 159/636 = 0.25
Xv = (Xw + 558) * 0.25
Yv = (Yw + 40) * 0.25

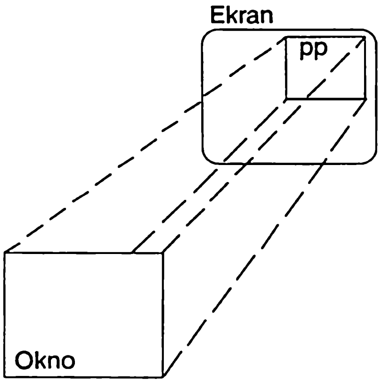
Punkt (0,200) we współrzędnych świata powinien być odwzorowany na punkt (139.5,60) we współrzędnych urządzenia, bowiem:

Xv = (0 + 558) * 0.25 = 139.5
Yv = (200 + 40) * 0.25 = 60.0

W wielu systemach, przed przesłaniem do wyświetlania będzie konieczne obcięcie współrzędnych urządzenia Xv i Yv do wartości całkowitych, w przeciwnym wypadku może powstać błąd. Dlatego też należy użyć wartości (139,60), zamiast (139.5,60.0).

Dla urządzenia o rozdzielczości 1280 × 1024, oknie o wymiarach –70 do 90, i 28 do 156 oraz polu prezentacji pokrywającym prawy górny kwadrat powierzchni ekranu (rys. 10.7) otrzymamy dane:

Vp = 1279 Wp = 90
Vl = 639 Wl = –70
Vg = 1023 Wxg = 156
Vd = 511 Wxd = 28
Xsk = (1279 – 639)/(90 + 70) = 640/160 = 4
Ysk = (1023 – 511)/(156 – 28) = 512/128 = 4



Rys. 10.7. Odwzorowanie okno/pole prezentacji

W tym przypadku punkt (25,100) w układzie współrzędnych świata będzie odwzorowany na punkt (1019,799), w układzie współrzędnych urządzenia, bowiem:

$$Xv = (25+70) * 4 + 639 = 1019$$
$$Yv = (100-28) * 4 + 511 = 799$$

Oba przykłady dobrano starannie tak, aby uniknąć zniekształcenia obrazu i żeby kształt okna zgadzał się z kształtem pola prezentacji; prowadzi to do równości Xsk = Ysk. Aby nie wystąpiły zniekształcenia (co jest czasami bardzo istotne) trzeba zawsze dobrać proporcje okna i pola prezentacji zgodne dla każdego rysunku. W celu ułatwienia prac projektowych, można używać tylko okien i pól prezentacji kwadratowych chyba, że wystąpią specjalne przyczyny, które zadecydują o innym wyborze. Pole prezentacji powinno być największym kwadratem, który można narysować na ekranie; boki kwadratu powinny być równe wysokości ekranu.

10.3. Pola prezentacji w znormalizowanych współrzędnych urządzenia

W poprzednim przykładzie określono współrzędne pola prezentacji przy założeniu rzeczywistych wymiarów urządzenia, dla którego należało przygotować rysunek. Jest to szybka i dobra metoda, jeżeli rysunek jest zawsze przedstawiany na tym samym urządzeniu. Jednak, jak to pokazano w rozdz. 6, jest wskazane pisać programy tak, żeby można było rysować na dowolnym urządzeniu. Niezależność od urządzenia uzyskuje się, wprowadzając do programu *znormalizowane współrzędne urządzenia* i *znormalizowane transformacje*. W tym przypadku metoda opisu okna jest taka sama jak poprzednio, natomiast wymiary pola prezentacji są określone w przedziale od 0 do 1, w obu kierunkach niezależnie od wymiarów i kształtów powierzchni do rysowania. Wyświetlenie obrazu na nowym urządzeniu wymaga jedynie przeskalowania obrazu po to żeby otrzymać właściwe wymiary.

Należy zatem napisać nowy program Ustaw-polePrezentacji, podobny do programu Ustaw-wsPolePrezentacji. Do testowania dopuszczalności zakresów, należy używać wartości: Xvmin = 0, Xvmax = 1, Yvmin = 0 i Yvmax = 1. Komunikat o wymiarach pola prezentacji powinien być zmieniony na "Pole prezentacji nie mieści się w kwadracie jednostkowym".
Poprzedni przykład przyjmie następującą postać:

Ustaw-okno (–90,90,28,156)
Ustaw-polePrezentacji (0.5,1.0,0.5,1.0)
WyznaczSkale
ZmianaWspółrz (Xw,Yw;Xv,Yv)

Programy WyznaczSkale i ZmianaWspółrz (które należało opracować jako zadania) powinny mieć postać taką, jak przedstawiono w tabl. 10.6 i tabl. 10.7. W wyniku działania obu programów otrzymujemy:

Xsk = 0.003125
Ysk = 0.003906
Xwzn = 0.796875
Ywzn = 0.781246

Tak więc punkt (25,100) we współrzędnych świata będzie punktem (0.796875, 0.781246) we współrzędnych znormalizowanych.

Tablica 10.6. Podprogram *WyznaczSkale*

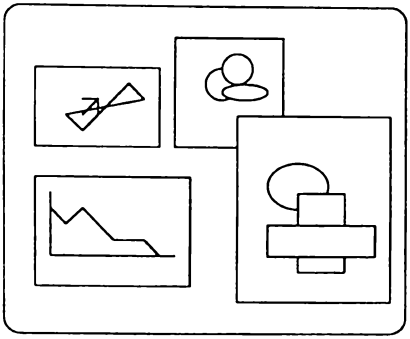
```
Subroutine WyznaczSkale
Global Integer Vl, Vp, Vd, Vg
Global Real Xsk, Ysk Wp, Wl, Wg, Wd
/* Podprogram ustala globalne współczynniki skali na podstawie parametrów
okna i pola prezentacji. Należy uwzględnić sprawdzanie, czy nie wystąpi dzielenie
przez zero, co może się zdarzyć wówczas, gdy Wp = Wl lub Wg = Wd */
        Xsk = (Vp – Vl)/(Wp – Wl)
        Ysk = (Vg – Vd)/(Wg – Wd)
End podprogramu WyznaczSkale
```

Tablica 10.7. Podprogram *ZmianaWspółrz*

```
Subroutine ZmianaWspółrz(Xwsp,Ywsp;Xwzn,Ywzn)
Real Xwsp, Ywsp, Xwzn, Ywzn
/* Podprogram zamienia współrzędne świata (Xwsp,Ywsp) na współrzędne znor-
malizowane (Xwzn, Ywzn). Vzl i Vzd są podane również we współrzędnych znor-
malizowanych */
Global Real Wl, Wd, Xsk, Ysk, Vzl, Vzd
        Xwzn = (Xwsp – Wl) * Xsk + Vzl
        Ywzn = (Ywsp – Wd) * Ysk + Vzd
End podprogramu ZmianaWspółrz
```

10.4. Wielokrotne pola prezentacji

Czasami chcemy uzyskać rysunki składające się z kilku okien i pól prezentacji na tej samej powierzchni przeznaczonej do wyświetlania, tak jak to pokazano na przykład na rys. 10.8. Można zrealizować to przez zmianę parametrów w podprogramach Ustaw-okno, Ustaw-polePrezentacji lub Ustaw-wsPolePrezentacji, albo też przez zdefinio-



Rys. 10.8. Wielokrotne pola prezentacji

wanie kilku okien i pól prezentacji na początku programu i wywoływanie ich wówczas, gdy zachodzi potrzeba. W pewnych okolicznościach np. przy wielokrotnych skokach od jednego okna do drugiego jest preferowana ta ostatnia procedura. Wymaga ona modyfikacji podprogramów tak, żeby można było określać kod liczbowy, rozróżniający kombinacje. Lista parametrów i sam podprogram powinny być zmienione tak, żeby możliwe było akceptowanie kodu i przechowanie rekordu różnych wartości. Fragment programu może przyjąć postać podaną w tabl. 10.8.

Tablica 10.8.

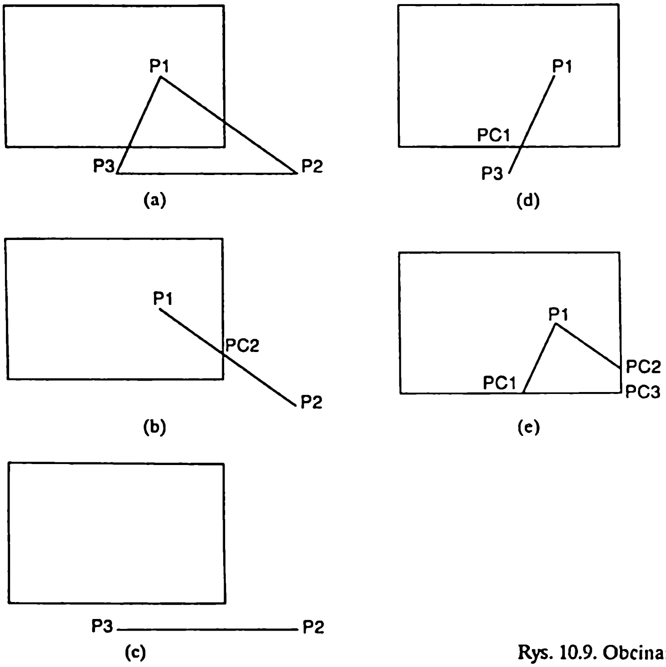
```
/* Fragment  programu z wieloma oknami i polami prezentacji */
Call Ustaw-okno(1,0,100,0,90)        /* okno 1 */
Call Ustaw-polePrezentacji(1,0,0.5,0,0.5)   /* pole prezentacji 1 */
Call Ustaw-okno(2,25,75,10,80)    /* okno 2 */
Call Ustaw-polePrezentacji(2,0.5,1.0,0.5,1.0)    /* pole prezentacji 2 */
        Call WyznaczSkale(1)
        Call ZmianaWspółrz(1)
        Call WyznaczSkale(2)
        Call ZmianaWspółrz(2)
/* Koniec fragmentu programu */
```

10.5. Obcinanie wielokątów

Wiele rysunków, które zechcemy wyświetlić będzie się składało raczej z wielokątów niż z pojedynczych linii. Jeżeli do programu obcinania zostanie przesłany zbiór linii two-

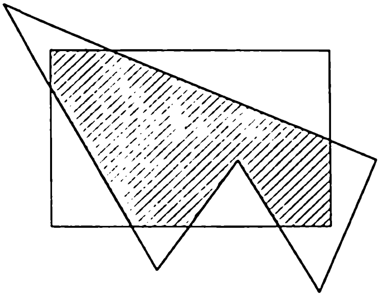
rzących wielokąt np. taki, jak na rys. 10.9a, to nastąpi:
- obcięcie odcinka P1-P2 w punkcie PC2 (rys. 10.9b),
- całkowite obcięcie odcinka P2-P3, który stanie się niewidoczny (rys. 10.9c),
- obcięcie odcinka P3-P1 w punkcie PC1 (rys. 10.9d).

To co zostało jest parą odcinków, a nie wielokątem. Nie stanowiłoby to problemu gdybyśmy rysowali tylko odcinki; jeżeli jednak chcemy uzyskiwać wielokąty (być może z myślą o późniejszym ich wypełnieniu), to musimy traktować obcięte figury tak, jak wielokąty. Oznacza to, że obcięty kształt z rys. 10.9a powinien być wielokątem o postaci jak na rys. 10.9e. W tym przypadku nowy wielokąt ma więcej boków i wierzchołków niż poprzedni. Tak zazwyczaj będzie i jak pokazano rys. 10.10 obcięty wielokąt może mieć dwa razy więcej boków niż wielokąt nieobcięty.

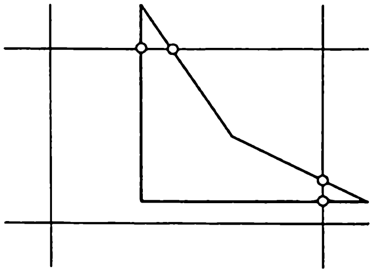


Rys. 10.9. Obcinanie wielokąta

W celu prawidłowego obcinania wielokątów należy wprowadzić nowy podprogram ObcinanieWielokąta. Różni się on znacznie od programu ObcinanieOdcinka i uwzględnia wielokąty o N wierzchołkach. Program wykorzystuje metodę opracowaną przez Sutherlanda i Hodgmana i działa na zasadzie testowania całego wielokąta względem kolejnych granic okna. Jeżeli wielokąt wychodzi na zewnątrz jednej z granic (przedłużonej do nieskończoności z każdej strony), to są dodawane nowe wierzchołki i jest tworzony nowy wielokąt w sposób pokazany na rys. 10.11. Ponieważ jest prawdopodobne, że w każdym kroku testowania będą dodawane nowe wierzchołki (a jest niemożliwe przewidzenie z góry ile takich wierzchołków będzie), to należy zapamiętać



Rys. 10.10. Obcięty wielokąt może mieć dwa razy tyle wierzchołków co nieobcięty wielokąt



Rys. 10.11. Nowe wierzchołki dodane wtedy, kiedy brzegi wielokąta przecinają granice obcinania

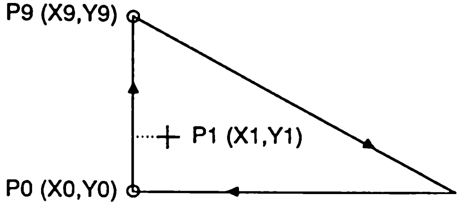
nowe wielokąty w tablicach, znacznie większych niż pierwotne. Metoda ta, nie jest więc efektywna pod względem wykorzystania pamięci komputera. Oryginalna metoda Sutherlanda i Hodgmana jest rekurencyjna i oszczędniej wykorzystuje pamięć. W książce podano prostą wersję podprogramu ObcinanieWielokąta, biorąc pod uwagę to, że w niektórych językach nie jest łatwo opracowywać programy rekurencyjne. W podprogramie ObcinanieWielokąta (tabl. 10.9), w przeciwieństwie do podprogramu ObcinanieOdcinka, nie jest konieczne przesuwanie okna do początku współrzędnych.

Tablica 10.9. Podprogram *ObcinanieWielokąta*

```
Subroutine ObcinanieWielokąta(N,Wielwe;NN,Wielwy)
Integer N, NN
Real Wielwe, Wielwy
/* Program obcina względem granic 1, 2, 3, 4 wielokąt o N wierzchołkach opisany
w Wielwe na wielokąt o NN wierzchołkach opisany w Wielwy. Wielwy musi mieć
rozmiar  (2 * N,2) */
Local Integer I, Licz, Granica, J, Wid1, Wid2, Wid3
Local Real Chwil, S1, S2, P1, P2, X, Y
/* Zdefiniuj lokalną tablicę Chwil i skopiuj do niej  Wielwe */
Dim Chwil(2 * N,2)
For I = 1 to N
        Chwil(I,1) = Wielwe(I,1)
        Chwil(I,2) = Wielwe(I,2)
Next I
For Granica = 1 to 4    /* dla każdej granicy /
        Licz = 1
        S1 = Chwil(N,1)   /* punkt S jest ostatnim punktem */
        S2 = Chwil(N,2)
        For J = 1 to N   /* dla każdego wierzchołka */
          P1 = Chwil(J,1)   /* punkt P jest następny */
          P2 = Chwil(J,2)
```

```
            Call Widoczny(Granica,P1,P2;Wid1)
            If (Wid1 = 1) Then    /* punkt P jest widoczny */
               Call Widoczny(Granica,S1,S2;Wid2)
               If (Wid2 = 1) Then    /* punkt S jest widoczny */
                  Wielwy(Licz,1) = P1
                  Wielwy(Licz,2) = P2
                  Licz = Licz + 1
               Else    /* punkt S jest niewidoczny */
                  Call Przecięcie(Granica,S1,S2,P1,P2;X,Y)
                  Wielwy(Licz,1) = X
                  Wielwy(Licz,2) = Y
                  Licz = Licz + 1
                  Wielwy(Licz,1) = P1
                  Wielwy(Licz,2) = P2
                  Licz = Licz + 1
               Endif
         Else    /* punkt P jest niewidoczny */
            Call Widoczność(Granica,S1,S2;Wid3)
            If (Wid3 = 1) Then
               Call Przecięcie(Granica,S1,S2,P1,P2;X,Y)
               Wielwy(Licz,1) = X
               Wielwy(Licz,2) = Y
               Licz = Licz + 1
            Endif
         Endif
/* Przejdź do następnego punktu */
            S1 = P1
            S2 = P2
         Next J
         NN = Licz - 1
         N = NN
/* Przypisz tablicę Chwil nowemu wielokątowi */
         For I = 1 to NN
            Chwil(1,1) = Wielwy(1,1)
            Chwil(1,2) = Wielwy(1,2)
         Next I
Next Granica
End podprogramu ObcinanieWielokąta
```

Podprogram korzysta z dwóch podprogramów: Przecięcie (tabl. 10.10), który określa punkt przecięcia odcinka z rozszerzoną granicą i Widoczność (tabl. 10.11), który decyduje, czy punkt jest wewnątrz granicy okna. "Wewnątrz", w tym kontekście oznacza, że punkt P1(X1,Y1) jest z prawej strony odcinka łączącego P0(X0,Y0)



Rys. 10.12. P1 znajduje się wewnątrz ponieważ zawsze jest położony z prawej strony odcinków (opis zgodny z ruchem wskazówek zegara)

i P9(X9,Y9) tak, jak pokazano na rys. 10.12. Warunek ten dla punktu P1 będzie spełniony, jeżeli V1 wyznaczone z niżej napisanej zależności jest dodatnie

$$V1 = (X9 - X0) * (Y9 - Y1) - (Y9 - Y0) * (X9 - X1)$$

Tablica 10.10. Podprogram *Przecięcie*

Subroutine Przecięcie(Granica,X1,Y1,X2,Y2;X,Y)
Integer Granica
Real X1, X2, X, Y1, Y2, Y
/*Podprogram określa współrzędne punktu (X,Y) przecięcia granicy przez odcinek łączący punkty (X1,Y1) i (X2,Y2) */
Global Real Wg, Wp, Wd, Wl
Local Real R
Case Granica **of**
  1 : R = (Wg – Y1)/(Y2 – Y1)
    X = X1 + (X2 – X1) * R
    Y = Wg
  2 : R = (Wp – X1)/(X2 – X1)
    Y = Y1 + (Y2 – Y1) * R
    X = Wp
  3 : R = (Wd – Y1)/(Y2 – Y1)
    X = X1 + (X2 – X1) * R
    Y = Wd
  4 : R = (Wl – X1)/(X2 – X1)
    Y = Y1 + (Y2 – Y1) * R
    X = Wd
Endcase
End podprogramu Przecięcie

Tablica 10.11. Podprogram *Widoczność*

Subroutine Widoczność(Granica,X1,Y1;V)
Integer Granica, V
Real X1, Y1
/*Podprogram ustawia V = 1, gdy punkt (X1,Y1) jest po widocznej stronie granicy 1, 2, 3 lub 4 */

```
Global Real Wg, Wp, Wd, Wl
Local Real V1
Case Granica of
1 : V1 = (Wp – Wl) * (Wg – Y1)
2 : V1 = – (Wd – Wg) * (Wp – X1)
3 : V1 = (Wl – Wp) * (Wd – Y1)
4 : V1 = – (Wg – Wd) * (Wl – X1)
Endcase
If (V1 < = 0) Then
        V = 0
Else
        V = 1
Endif
End podprogramu Widoczność
```

ZADANIA

10.1. Jeżeli rozwiązałeś zadanie 2.4, to wiesz, jak system się zachowuje w przypadku próby rysowania poza polem wyświetlania. Może się okazać, że komputer sam realizuje obcinanie i ma własną wersję podprogramów okna i pola prezentacji. Jeżeli tak jest, spróbuj różnych kombinacji okno/pole prezentacji, żeby poznać występujące efekty. W szczególności spróbuj określić bardzo duże okna, np. o szerokości większej niż 32000 jednostek. Czy system radzi sobie z tym przypadkiem?

10.2. Jeżeli system poprawnie realizuje obcinanie rysunków na ekranie, to sprawdź, czy realizuje on również obcinanie na urządzeniach zewnętrznych, takich jak plotery. Mogą wystąpić dwie możliwości:
- obcinanie jest wykonywane w komputerze i tylko obcięta informacja jest wysyłana do urządzenia,
- urządzenie zewnętrzne samo wykonuje obcinanie. Jeżeli urządzenie jest dołączone do komputera za pomocą wolnego łącza szeregowego i podejmuje próbę wykonania rysunków, które trzeba obcinać, to może się okazać, że lepiej jest wykorzystać własny program obcinania, w celu zminimalizowania ilości "niewidocznych" danych, które są przesyłane linią. W istocie, chodzi o kompromis między czasem obliczeń a czasem przesyłania. Do wykazania spodziewanych oszczędności (jeśli w ogóle wystąpią) należy: zrealizować podprogramy podane w rozdziale i bezpośrednio znaleźć różnice, albo wykonać odpowiednie obliczenia.

10.3. Zakoduj podane podprogramy i użyj ich do sprawdzenia przykładów podanych w tekście. Spróbuj umieścić pole prezentacji w różnych punktach powierzchni wyświetlania.

10.4. Wykorzystaj współrzędne litery podane w rozdz. 7 do sprawdzenia, jak różne kombinacje okno/pole prezentacji zmieniają kształt i proporcje.

10.5. Podprogram WyznaczSkale podany w tabl. 10.6 nie będzie działał poprawnie i doprowadzi do błędu systemowego, jeżeli zostanie wywołany wówczas, gdy

Wp = Wl albo Wg = Wd. Dlaczego tak jest? W podprogramie umieść test zapobiegający takiej możliwości.

10.6. Opracuj metodę, która pozwoli zachować ten sam współczynnik proporcji między oknem a polem prezentacji tak, żeby zachować proporcje.

10.7. Sprawdź podprogramy dla wielu okien i pól prezentacji na tej samej powierzchni wyświetlania. Uwzględnij pola prezentacji nakładające się i rozłączne.

10.8. Po zakodowaniu różnych podprogramów obcinania wielokąta, sprawdź je korzystając z różnych kształtów. Uwzględnij wielokąty całkowite zawierające w sobie okno oraz takie, które zawierają się całkowicie w oknie. Napisz podprogramy w kodzie maszynowym, w celu przyspieszenia ich działania.

10.9. Napisz i przetestuj program Ustaw-polePrezentacji tak, żeby mógł działać we współrzędnych znormalizowanych zgodnie z opisem na początku rozdziału. Wypróbuj różne kombinacje okien i pól prezentacji w znormalizowanych jednostkach i pól prezentacji określanych w jednostkach urządzenia. Wymień zalety i wady możliwości korzystania z tego dodatkowego kroku w procesie rysowania.

11. Wyświetlanie

Po określeniu modelu, dokonaniu transformacji i obcięciu, obiekty dwuwymiarowe są przygotowane do wyświetlenia. Do wyświetlania zostaną wykorzystane instrukcje graficzne wprowadzone w rozdz. 2. Rzeczywisty wygląd rysunków zależy od dostępnych urządzeń wyjściowych i oprogramowania; można zatem rysować:

- obrazy składające się z odcinków lub cieniowane,
- obrazy czarno-białe lub kolorowe,
- linie schodkowe lub gładkie itp.

Ponadto można wyświetlać obrazy częściami lub w całości, interakcyjne lub nieinterakcyjne, jako pojedyncze obrazy lub jako serie, które utworzą sekwencję animowaną.

11.1. Rysowanie odcinków

Niektóre urządzenia, np. plotery lub lampy pamięciowe mogą być wykorzystywane w wygodny sposób tylko do rysowania odcinków. Na takich urządzeniach można wyświetlać obrazy złożone z oddzielnych odcinków i utworzone za pomocą prostych programów takich jak RysowanieOdcinka. W programach stosuje się instrukcje graficzne MOVE i DRAW (tabl. 11.1).

Każdy z N odcinków wchodzących w skład obrazu, może być utworzony za pomocą fragmentu programu pokazanego w tabl. 11.2. W wyniku działania tego pro-

Tablica 11.1. Podprogram *RysowanieOdcinka*

```
Subroutine RysowanieOdcinka(Xst,Yst,Xkon,Ykon)
Integer Xst, Yst, Xkon, Ykon
/* Podprogram rysuje odcinek łączący punkty (Xst,Yst) i (Xkon,Ykon). Oba punkty
są określone we współrzędnych urządzenia */
        MOVE (Xst, Yst)
        DRAW (Xkon, Ykon)
End podprogramu RysowanieOdcinka
```

Tablica 11.2.

```
/* Fragment programu rysowania N oddzielnych odcinków */
        Ustaw-okno(Wxmin,Wxmax,Wymin,Wymax)
        Ustaw-wsPoleWidzenia(Vxmin,Vxmax,Vymin,Vymax)
        WyznaczSkale    /* oblicz Xsk i Ysk */
        For J = 1 to N
          Read X1, Y1, X2, Y2
/* Tu mogą być transformacje */
          Call ObcinanieOdcinka(X1,Y1,X2,Y2;X1n,Y1n,X2n,Y2n,Wid)
          If Wid = 1 Then   /* zamień współrzędne świata
                              na współrzędne urządzenia */
            Call ZamianaWspółrz(X1n,Y1n;X1v,Y1v)
            Call ZamianaWspółrz(X2n,Y2n;X2v,Y2v)
            Call RysowanieOdcinka(X1v,Y1v,X2v,Y2v)
          Endif
        Next J
/* Koniec fragmentu programu */
```

gramu można uzyskiwać zbiór rozłącznych odcinków (np. siatki) lub połączonych odcinków (np. wielokąty). Każde wywołanie podprogramu RysowanieOdcinka powoduje wykonanie instrukcji MOVE i DRAW. Nie jest to konieczne, jeżeli koniec jednego odcinka jest początkiem następnego, tak jak w wielokącie.

Jest prawdopodobne, że większość rysunków będzie tworzona z wielokątów; więc jest potrzebny podprogram, który pozwoli rysować linie ciągłe. Przy założeniu, że współrzędne końców tych odcinków są przechowywane w dwóch, N-elementowych tablicach Xtab i Ytab, można utworzyć podprogram Łamana (tabl. 11.3).

Do utworzenia rysunku złożonego z wielokątów będzie potrzebny fragment programu taki, jak przedstawiono w tabl. 11.4.

Tablica 11.3. Podprogram *Łamana*

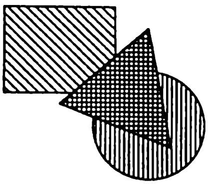
```
Subroutine Łamana(N,Xtab,Ytab)
Integer N, Xtab, Ytab
/* Podprogram rysuje linię ciągłą łamaną określoną przez N punktów. Często
punkt N pokrywa się z punktem 1. Dla N = 2 jest rysowany odcinek */
Local Integer J
        MOVE (Xtab(1),Ytab(1))   /* przejdź do pierwszego punktu */
/* Rysuj do każdego następnego punktu */
        For J = 2 to N
          DRAW (Xtab(J), Ytab(J))
        Next J
End podprogramu Łamana
```

Tablica 11.4.

```
/* Fragment programu rysowania wielokątów */
        Call Ustaw-okno(Wxmin,Wxmax,Wymin,Wymax)
        Call Ustaw-wsPolePrezentacji(Wxmin,Wxmax,Wymin,Wymax)
        Call WyznaczSkale    /* oblicz Xsk i Ysk */
        For Wielokąt = 1 to Makswiel
           Read N
           For J = 1 to N
              Read Xtab(J), Ytab(J)
        Next J
/* Tu mogą być pewne transformacje */
        Call ObcinanieWielokąta(N,Xtab,Ytab;Nowen,Xwy,Ywy)
/* Tu zmień współrzędne świata na współrzędne urządzenia */
        Call Łamana(Nowen,Xwy,Ywy)      /* rysuj łamaną */
        Next Wielokąt
/* Koniec fragmentu programu */
```

11.2. Cieniowanie

Rysowanie odcinków jest atrakcyjnym i użytecznym sposobem przekazywania informacji graficznej. Jednak często zachodzi potrzeba wypełnienia powierzchni w celu poprawienia wyglądu rysunków. Korzystając z urządzeń wektorowych takich jak ploter czy lampa pamięciowa, wypełnienie można uzyskać przez zakreskowanie pola za pomocą jednakowo oddalonych od siebie odcinków tak, jak pokazano na rys. 11.1. W urządzeniach rastrowych, można całkowicie wypełnić powierzchnię, rozświetlając wszystkie należące do niej piksele.



Rys. 11.1. Kreskowanie za pomocą odcinków równo oddalonych od siebie

Dla wielokąta opisanego dwoma tablicami Pxtab(N) i Pytab(N), kreskowanie uzyskuje się za pomocą procedury Kreskowanie (tabl. 11.5), która uwzględnia współrzędne urządzenia. Funkcje Min i Max, wywoływane w tym programie są niedostępne w niektórych językach. Można je symulować w następujący sposób:

Min(X,Y) = Abs(X < = X) * X + Abs(X > Y) * Y

Max(X,Y) = Abs(X > = Y) * X + Abs(X < Y) * Y

Wartości bezwzględne (Abs) są potrzebne, ponieważ w niektórych systemach dla

TRUE używa się –1, a dla FALSE 0. Jeżeli w danym systemie dla TRUE używa się 1 a dla FALSE 0, to Abs można pominąć. Jeśli urządzenie wyświetlające wymaga tego, to P1 powinno być zaokrąglone do wartości całkowitej, w ostatniej części programu. W urządzeniach rastrowych cały wielokąt można wypełnić, wykorzystując w tym celu podprogram kreskowania. Stosuje się kreskowanie poziome, w każdej linii rastra. Należy zatem obliczać przecięcia ciasno upakowanych linii z brzegami wielokąta, co sprawia, że korzystanie z podprogramu w obecnej postaci będzie bardzo wolnym procesem. Można go przyspieszyć wykorzystując fakt, że odcinki są poziome i obliczanie przecięć jest całkiem proste, nie wymaga bowiem obracania wielokąta. Można zatem w podprogramie pominąć część związaną z obrotami.

Tablica 11.5. Podprogram *Kreskowanie*

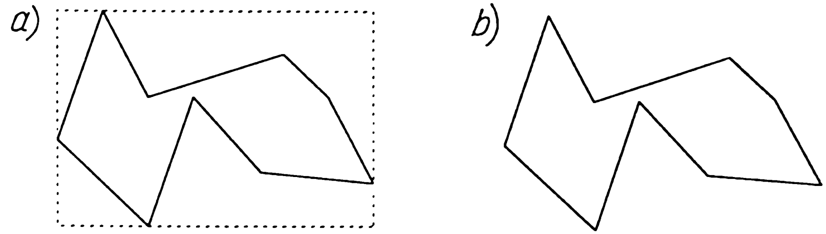
```
Subroutine Kreskowanie(N,PA,A,D)
Integer N, PA, D
Real A
/* Podprogram wypełnia zamknięty wielokąt PA o wymiarach (N, 2) odcinkami
odległymi od siebie o D jednostek urządzenia, prowadzonymi pod kątem A°.
Ponieważ jest to program wyświetlania, wykorzystuje on współrzędne urządzenia.
Odcinki wypełniające nie są pamiętane */
Local Integer JJ, J, K
Local Real P1, Co, L, S, Si, W, W1, W2, X, X1, Y2
      Dim W(N,2),X(N),P1(2)
      S = 100000
      L = –100000
      A = A * 3.14159 / 180    /* zamień na radiany */
      Si = Sin(A)
      Co = Cos(A)
      For JJ = 1 to N    /* obróć wielokąt o –A */
         W(JJ,1) = PA(JJ,1) * Co + PA(JJ,2) * Si
         W(JJ,2) = PA(JJ,2) * Co – PA(JJ,1) * Si
         S = Min(S, W(JJ,2))
         L = Max(L, W(JJ,2))
      Next JJ
      Y2 = S + 0.5 * D
      While Y2 < = L Do
         K = 0
         For JJ = 2 to N
            W1 = Min(W(I–1,2),W(JJ,2))
            If Y2 > = W1 Then
               W2 = Max(W(I–1,2),W(JJ,2))
               If Y2 < W2 Then
                  X1 = (W(I–1,2)–Y2) * (W(I–1,1)–W(JJ,1))
                  X1 = W(I–1,1)–X1/(W(I–1,2)–W(JJ,2))
```

```
                K = K + 1
                For J = K to 1 Step - 1
                   If J = 1 Then
                      X(J) = X1
                   Else
                      If X(J-1) = > X1 Then
                         X(J) = X1
                         J = 1
                      Else
                         X(J) = X(J-1)
                      Endif
                   Endif
                Next J
             Endif
          Endif
       Next JJ
       For JJ = 1 to K
          P1(1) = X(JJ) * Co - Y2 * Si /* odwrotny obrót */
          P1(2) = X(JJ) * Si + Y2 * Co
          If 2 * Int(0.5 * JJ) - JJ >- 0.1 Then
             DRAW(P1(1),P1(2))  /* jeżeli JJ nieparzyste */
          Else
             MOVE(P1(1),P1(2))  /* jeżeli JJ parzyste */
          Endif
        Next JJ
        Y2 = Y2 + D   /* następny odcinek */
       Endwhile
End podprogramu Kreskowanie
```

W większości systemów rastrowych stosuje się inną metodę cieniowania. Programowo należy sprawdzić, czy określony piksel jest wyświetlany; dzięki temu można łatwo wykrywać brzegi wielokąta. Zauważmy bowiem, że jeżeli poruszamy się wzdłuż linii rastra i napotkamy już wyświetlony piksel, to przechodzimy z zewnętrznej strony wielokąta do jego wnętrza i odwrotnie. A więc startując z wnętrza wielokąta, w określonej linii rastra, można zacząć wyświetlać kolejne piksele z prawej i z lewej strony dopóki nie napotkamy piksela już wyświetlonego co oznacza, że należy się zatrzymać. Następnie przechodzimy do linii leżących powyżej i poniżej i powtarzamy procedurę, aż do wypełnienia całego wielokąta.

Ta prosta metoda jest poprawna dla wielokątów wypukłych takich jak trójkąty, prostokąty i okręgi. W zastosowaniu do bardziej skomplikowanych figur, np. takich jak przedstawiona na rys. 11.2a, metoda zawodzi. Dla takich figur, określona linia rastra może przecinać brzeg figury wiele razy. Należy przyjąć następującą metodę

cieniowania. Dla figury określamy maksymalne i minimalne wartości X i Y oraz wyznaczamy prostokąt opisany na figurze (rys. 11.2b). Następnie, dla dowolnej linii rastra włączamy piksele jeśli jesteśmy wewnątrz wielokąta i nie osiągnęliśmy brzegu otaczającego prostokąta; dojście do brzegu prostokąta oznacza, że nie ma więcej krawędzi do przejścia.



Rys. 11.2. Obiekt: a) z opisanym prostokątem; b) dowolny

Obie metody wymagają oczywiście, żeby brzegi wielokąta były narysowane wcześniej i nie będą działały poprawnie, jeżeli wystąpi nawet jednopikselowa przerwa w linii. W drugiej metodzie powstanie dodatkowy problem wpływu innych części rysunku, tych, na które zachodzi opisany prostokąt. Można temu zaradzić jeżeli znajdziemy sposób określenia, które z dotychczas wyświetlanych pikseli należą do cieniowanego wielokąta. Jest to zazwyczaj możliwe, jeżeli piksele należące do różnych wielokątów mają różne kolory.

Ponieważ wypełnianie wielokąta jest bardzo potrzebne, opracowano wiele algorytmów do efektywnej realizacji wypełniania w technice rastrowej. Są to algorytmy dwóch rodzajów: wypełnianie wielokąta po uprzednim określeniu brzegów wielokąta i koloru wypełniania oraz wypełnianie typu malowanie. W drugiej metodzie wypełniania dodatkowo jest konieczne określenie punktu leżącego wewnątrz wielokąta, od którego rozprzestrzenia się wzór wypełniający wielokąt. Ta metoda jest wykorzystywana w systemach malarskich, ponieważ łatwo jest w nich wskazać początkowy piksel.

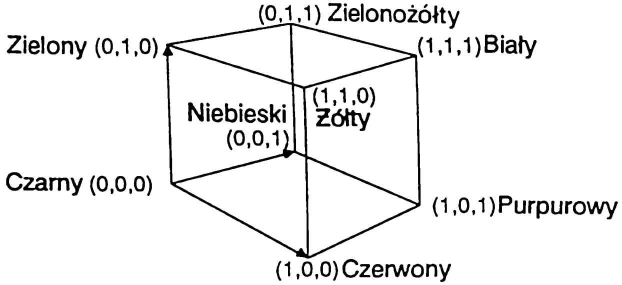
Jeżeli istotna jest szybkość, to algorytmy wypełniania powinny być programami pisanymi w kodzie maszynowym. Obecnie wielu producentów włącza procedury automatycznego wypełniania wielokąta do firmowego oprogramowania swoich urządzeń rastrowych. Należy oczekiwać, że ta tendencja będzie się utrzymywała.

11.3. Rysowanie w kolorze

Większość systemów graficznych wykorzystujących technologię rastrową, pozwala tworzyć rysunki wielokolorowe. W komputerach domowych liczba dostępnych kolorów jest ograniczona do 8 lub 16 i zestaw ich jest narzucony przez producenta. W tych ograniczonych systemach, kolory mają przypisane nazwy lub numery. Wyboru doko-

nuje się za pomocą takich poleceń jak: COLOUR = RED; HCOLOR = 5; COLOR3,2, lub podobne. W wielu profesjonalnych systemach, a nawet w niektórych komputerach personalnych, jest możliwy znacznie szerszy wybór umożliwiający realizację określonych potrzeb. Takie systemy pozwalają na definiowanie kolorów za pomocą procentowego udziału kolorów: czerwonego, zielonego i niebieskiego. Tak więc do zdefiniowania koloru 15, będącego mieszaniną kolorów czerwonego w 70% i zielonego w 30%, można użyć instrukcji Setcolour (15,0.7,0.3,0).

Określając kolory przez specyfikację procentowego udziału kolorów: czerwonego, zielonego i niebieskiego, wykorzystujemy tzw. *model kolorów RGB* (ang. R - red, G - *green*, B - *blue*). Model RGB można traktować jako system trójwymiarowy, w którym podstawowe kolory zmieniają się od 0 do 1 na osiach, a wszystkie możliwe kolory leżą wewnątrz jednostkowego sześcianu (rys. 11.3).



Rys. 11.3. Sześcian RGB

W początku układu (0,0,0) znajduje się kolor czarny, bowiem udział każdego z podstawowych kolorów jest tu zerowy. Punkt (1,1,1) odpowiada kolorowi białemu; w pełni są tu wykorzystywane kolory składowe. Na przekątnej od początku układu (0,0,0) do punktu (1,1,1) znajdują się odcienie szarości.

W systemach rastrowych nie jest możliwe pokazanie wszystkich kolorów, które możemy obserwować w naturze. Światło jest emitowane przez pobudzony luminofor, który ma swoje ograniczenia w odtwarzaniu koloru. Sześcian kolorów pokazuje jedynie zakres kolorów, jakie określony system może wytworzyć. Trzeba również podkreślić, że chociaż można określać ilości pierwotnych kolorów w ciągłym zakresie od 0 do 1, to cyfrowe sterowanie wyrzutnią elektronów (kolorów) sprawia, że zmiany występują skokowo: im mniejszy skok, tym więcej jest dostępnych kolorów. W związku z tym niektóre systemy wymagają podawania wartości RGB w postaci liczb całkowitych z zakresu od 0 do 255, inne z zakresu od 0 do 4095 itd.

Nie jest łatwo dobrać odpowiednią kombinację składowych: czerwonej, zielonej i niebieskiej do uzyskania określonego koloru, który sobie wyobrażamy. Oto kilka wskazówek ułatwiających tę pracę:

- punkt leżący na jednej z osi daje odcień koloru pierwotnego, im bliżej znajduje się początku układu współrzędnych tym odcień jest ciemniejszy;

- punkty leżące na płaszczyźnie wyznaczonej przez dwie osie reprezentują mieszaniny dwóch barw podstawowych. Mieszanina koloru czerwonego i zielonego ma tendencję do zabarwienia żółtego. Mieszanina koloru czerwonego i niebieskiego jest zbliżona do kolorów purpurowych. Zmieszanie koloru niebieskiego i zielonego daje w wyniku kolory określane jako niebieskozielone;
- mieszaniny trzech podstawowych barw mają tendencję do barwy szarej z odcieniem wyznaczonym przez dominującą barwę pierwotną. Przy jednakowych ilościach trzech barw podstawowych powstaje neutralna barwa szara;
- barwy umieszczone na zewnętrznych powierzchniach sześcianu oddalonych od początku układu współrzędnych są określane jako odcienie i zawierają pewien procent koloru białego.

W niektórych systemach są stosowane alternatywne metody specyfikowania barw. Jedna z nich wywodzi się z *modelu HLS* (ang. *hue* - barwa, *lightness* - jasność, *saturation* - nasycenie). Model ten jest uważany za bardzo udaną koncepcję.

Od dziewiętnastego wieku wiadomo, że jesteśmy wrażliwi na trzy różne składniki koloru. Są to:

- **Barwa** - jest to dominująca długość fali świetlnej i element, który określa nazwę koloru - czerwony, zielony, fiolet itd. Światło o dominującej długości fali, np. 578 nm jest nazywane żółtym, a o długości fali 630 nm czerwonym.
- **Intensywność** lub **jasność** (będziemy używać tych określeń jako synonimów, mimo że ich znaczenia nieco się różnią) - jest to strumień świetlny docierający do oka.
- **Nasycenie** - jest to stopień rozcieńczenia barwy przez biel. Kolor, który wydaje się nie zawierać bieli jest określany jako nasycony. Nasycenie koloru jest najbardziej związane z liczbą długości fal tworzących ten kolor; im liczba jest mniejsza, tym bardziej kolor jest nasycony.

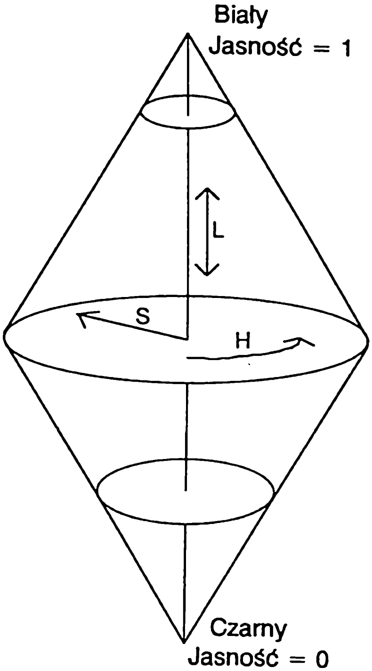
Powyższe elementy występują w modelu HLS, który umożliwia niezależne określenie każdego z elementów wpływających na naszą percepcję koloru. Graficznie model HLS jest reprezentowany przez podwójny stożek, taki jak pokazano na rys. 11.4.

Różne barwy są rozmieszczone wokół kół będących poziomymi przekrojami stożków. W niektórych systemach barwa jest określona za pomocą miary kątowej (od 0° do 360°); kolorowi niebieskiemu jest przypisany kąt 0°, czerwonemu 120° i zielonemu 240°. W innych systemach skala zmienia się od 0 do 1; kolorowi niebieskiemu jest przypisane 0, czerwonemu 0,333 i zielonemu 0,666. Barwa dopełniająca dla każdego koloru znajduje się po przeciwnej stronie okręgu.

Nasycenie (S) jest reprezentowane przez odległość wzdłuż promienia dowolnego koła barwy; pełne nasycenie (o wartości 1) występuje na obwodzie.

Oś jasności (L) przechodzi przez wierzchołki stożków. Dla koloru czarnego jasność jest równa 0, a dla białego jasność jest równa 1. Korzystając z tego systemu można określić kolor, np. jasnoróżowy będzie opisany jako $H = 120$, $L = 0{,}7$ i $S = 0{,}9$.

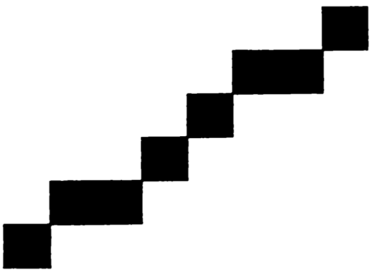
Jakiejkolwiek używalibyśmy metody opisania kolorów (znane są i inne metody) to jest prawdopodobne, że sam system będzie wykorzystywał tablicę odwzorowania kolorów (rozdz. 4) do przechowywania wartości RGB dopuszczalnych do wyświetlania.



Rys. 11.4. Model HLS

11.4. Linie schodkowe i gładkie

W systemach rastrowych, odcinki nachylone pod pewnym kątem mają wygląd zniekształcony, co jest skutkiem utworzenia ich z ciągów pojedynczych punktów, z których każdy można traktować jako małą "próbkę" rzeczywistego odcinka. W większości systemów rastrowych, do rysowania odcinków wykorzystuje się *algorytm Bresenhama*, opracowany na początku ery rozwoju grafiki komputerowej. W tym pomysłowym algorytmie wykorzystuje się tylko arytmetykę całkowitoliczbową (dzięki czemu algorytm łatwo realizuje się w kodzie maszynowym) do wyznaczenia punktów leżących najbliżej prawdziwego odcinka. Na rysunku 11.5 pokazano fragment odcinka o nachyleniu 5/7.

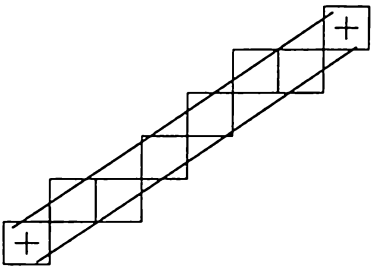


Rys. 11.5. Odcinek o nachyleniu 5/7 narysowany za pomocą algorytmu Bresenhama

Metoda eliminacji schodkowego wyglądu linii polega na wyznaczaniu punktów, które są na tyle małe i są tak blisko siebie, że oko nie może odróżnić ich od prawdziwego odcinka. Oznacza to konieczność używania monitorów o 4000 linii, a nie 500 lub 1000 linii, które są obecnie używane. Jednakże nikt nie produkuje takich monitorów, a nawet gdyby były dostępne, to wymagane szybkości przetwarzania danych stwarzałyby kolejne trudności.

Rozwiązaniem problemu "schodkowego" jest metoda rysowania odcinków, nie jako zbioru pojedynczych punktów o takiej samej jasności, ale jako grupy punktów o stopniowanej jasności lub poziomach szarości. W wyniku otrzymuje się trochę grubsze i postrzępione linie, które przy obserwacji z normalnej odległości widzenia są odbierane jako ostre, proste linie. Metoda ta jest określana jako *odkłócanie*. Do efektywnego odkłócania jest potrzebne minimum 16 poziomów szarości od białego do czarnego lub 16 kolorów od maksymalnej do minimalnej jasności.

Jeżeli popatrzymy na rys. 11.6, to zauważymy, że linia o szerokości jednego piksela biegnąc wzdłuż siatki zajmuje (procentowo) różne części powierzchni poszczególnych pikseli. Stwierdzono, że można uzyskać linie, w których zakłócenia są ukryte jeżeli jasność każdego piksela jest proporcjonalna do powierzchni zajmowanej



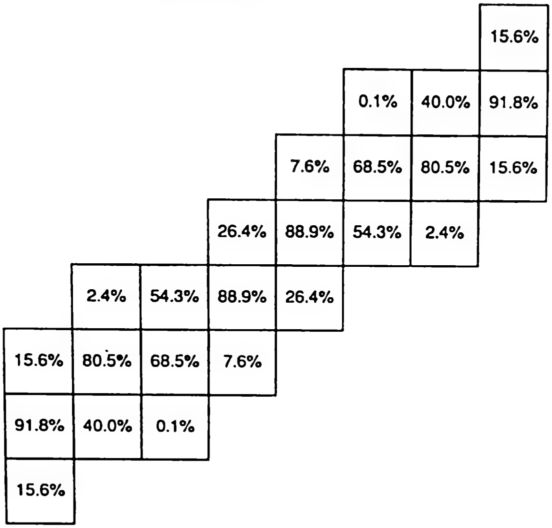
Rys. 11.6. Linia prosta zajmuje różne części powierzchni pikseli

przez linię. Na rysunku 11.7 pokazano wartości procentowe dla linii o nachyleniu 5/7.

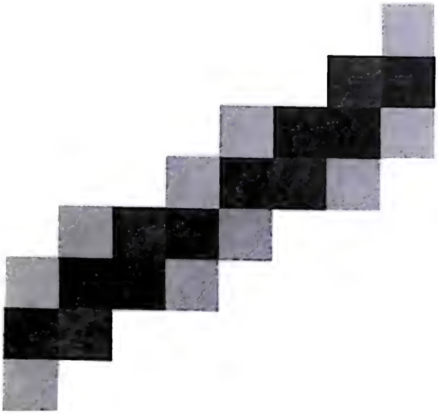
Obliczanie procentów jest stosunkowo wolnym zadaniem i dlatego opracowano wiele metod minimalizacji tego procesu - w tym modyfikacje algorytmu Bresenhama. Niektóre nowe modele monitorów mają wbudowane, sprzętowe mechanizmy odkłócające i taki sprzęt będzie się upowszechniał w miarę upływu czasu.

Na rysuku 11.8 pokazano fragment linii odkłóconej dzięki wykorzystaniu różnych poziomów szarości.

Na monitorze rastrowym szczególnie silnie są zakłócone okręgi. W systemach, które umożliwiają rysowanie okręgu za pomocą polecenia, stosuje się raczej wersję algorytmu Bresenhama niż metodę aproksymacji wielokątem, którą przedstawiono w rozdz. 8. Typowym przykładem podprogramu rysowania okręgu jest podprogram opracowany przez polskiego naukowca - informatyka profesora Dorosa. Podprogram Doros (tabl. 11.6) wykorzystuje współrzędne urządzenia i generuje nieodkłócony okrąg, który jest tworzony z ośmiu oddzielnych oktantów. Program ten daje najlepsze



Rys. 11.7. Procentowa zajętość powierzchni pikseli, przez które przechodzi linia prosta (zwróćmy uwagę na duży stopień symetrii)



Rys. 11.8. Fragment linii prostej odkłóconej przez użycie trzech poziomów szarości

przybliżenie w liczbach całkowitych; korzysta z podprogramu Punkt, który z kolei wykorzystuje RysowanieOdcinka do kreślenia punktu (tabl. 11.7).

Tablica 11.6. Podprogram *Doros*

Subroutine Doros(Xc,Yc,R)
Real Xc, Yc, R
/* Podprogram rysuje najlepszą całkowitą aproksymację okręgu o promieniu R i środku w punkcie (Xc,Yc). Procedura oblicza i rysuje punkty oktant po oktancie */
Local Integer X, X0, Y, Y0, Okt, J

```
X0 = Int(Xc + 0.5)   /* zaokrąglaj parametry do najbliższej wartości całkowitej */
        Y0 = Int(Yc + 0.5)
        S = Int(R + 0.5)
        X = R           /* ustalenie warunków początkowych */
        Y = 0
        For Okt = 1 to 8
          If Okt = Int(Okt/2) * 2 Then     /* Okt jest parzyste */
            S = S + 2 * X
            While Y > = 0 Do
              S = S - 2 * Y + 1
              Y = Y - 1
              If S < 0 Then
                S = S + 2 * X + 2
                X = X + 1
              Endif
              If X < > Then /* punkt nie leży na przekątnej */
                Call Punkt(Okt,X0,Y0,X,Y,)
              Endif
            Endwhile
          Else          /* Okt jest nieparzysty */
            S = S - 2 * X
            While Y < = X Do
              Call Punkt(Okt,X0,Y0,X,Y)
              S = S + 2 * Y + 1
              Y = Y + 1
              If S > = 0 Then
                S = S - 2 * X + 2
                X = X - 1
              Endif
            Endwhile
          Endif
        Next Okt
End podprogramu Doros
```

Tablica 11.7. Podprogram *Punkt*

```
Subroutine Punkt(Okt,X0,Y0,X,Y,)
Integer Okt, X0, Y0, X, Y
/* Podprogram jest wywoływany z podprogramu Doros z zadaniem rysowania
punktu, metodą rysowania odcinka o początku i końcu w tym samym punkcie.
Jeżeli dany system daje możliwość rysowania punktu to należy z niej korzystać
zamiast tego podprogramu */
Case of Okt
```

```
        1: Call RysowanieOdcinka(X0 + X,Y0 + Y,X0 + X,YO + Y)
        2: Call RysowanieOdcinka(X0 + Y,Y0 + X,X0 + Y,YO + X)
        3: Call RysowanieOdcinka(X0 - Y,Y0 + X,X0 - Y,YO + X)
        4: Call RysowanieOdcinka(X0 - X,Y0 + Y,X0 - X,YO + Y)
        5: Call RysowanieOdcinka(X0 - X,Y0 - Y,X0 - X,YO - Y)
        6: Call RysowanieOdcinka(X0 - Y,Y0 - X,X0 - Y,YO - X)
        7: Call RysowanieOdcinka(X0 + Y,Y0 - X,X0 + Y,YO - X)
        8: Call RysowanieOdcinka(X0 + X,Y0 - Y,X0 + X,YO - Y)
End podprogramu Punkt
```

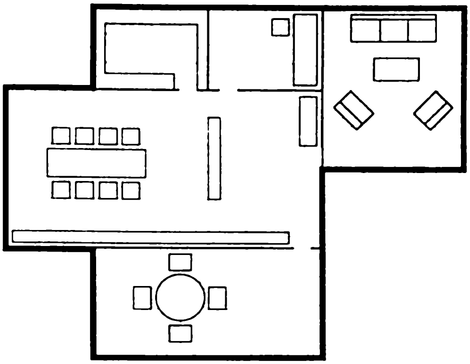
11.5. Segmenty i struktury danych

Dotychczas używaliśmy oddzielnych tablic do opisania elementów, które chcieliśmy narysować. Czasami jedna tablica wystarczała do opisania obiektu, kiedy indziej potrzebna była większa ich liczba. Wówczas, gdy chcemy tylko pokazać obiekty na rysunku, można dokonywać transformacji, obcinania i wyświetlania tablic, jedna po drugiej, aż do skompletowania rysunku. Procedurę tę można podsumować następująco:

Procedura Wyświetlania 1
  Gdy są jeszcze dane zrób co następuje:
    Czytaj liczbę N współrzędnych spodziewanych punktów;
    Czytaj N par współrzędnych do tablicy;
    O ile konieczne, dokonaj transformacji tablicy;
    Obetnij każdy odcinek, który wychodzi poza obszar
        ekranu i utwórz z reszty wielokąty;
    Wyświetl te wielokąty.

W systemach z małą pamięcią operacyjną i bez pamięci dyskowej, jest to prawdopodobnie najlepsze rozwiązanie procedury tworzenia i wyświetlania rysunku. Do umieszczenia danych opisu są potrzebne tylko dwie lub trzy tablice dostępne w dowolnej chwili; można z nich stale korzystać przy realizacji rysunków o dużej złożoności. Przy wyświetlaniu rysunków tym sposobem, nie ma miejsca na interakcję. Mimo, że można budować opis interakcyjnie, to komputer nie ma informacji o strukturze rysunku tworzonego tą metodą. Dlatego też, nie można dokonać edycji rysunku bez zmiany danych i ponownego wykonania programu. Zatem jest konieczna bardziej ogólna struktura danych niż ta, którą dotychczas proponowano.

Ponadto są zastosowania, dla których należy tworzyć i wyświetlać rysunki składające się z logicznych części, z których każda reprezentuje pewną odrębną całość. Na przykład rysunki architekta mogą składać się z: siatki o skoku 1 m^2 do markowania wymiarów; linii ukazujących zewnętrzne i wewnętrzne ściany; prostokątów i innych kształtów oznaczających meble itd. (rys. 11.9). Czasami jest konieczne pokazanie wszystkich tych elementów; kiedy indziej jest celowe pokazanie tylko ścian i siatki bez mebli, lub po prostu samych ścian. Alternatywnie, w animacji można utrzymać nieruchome tło, a posuwać i obracać obiekty znajdujące się bliżej obserwatora. Te procesy selektywnego wyświetlania są łatwiejsze, jeżeli opiszemy obiekty jako *segmenty*



Rys. 11.9. Rysunek architektoniczny

- części, które mogą być traktowane jako oddzielne, identyfikowalne jednostki. Idea polega na tym, że segmenty mogą być wyświetlone (lub nie), transformowane, usuwane lub też może być zmieniona ich nazwa bez wpływu na inne części rysunku. Zakres działań na segmentach zależy od cech funkcjonalnych urządzenia wyświetlającego. Na przykład, w lampach pamięciowych lub w ploterach można tylko dodawać segmenty do już przedstawionego rysunku. Usunięcie lub zmiana położenia lub orientacji, wymaga przerysowania całego obrazu. Nawet w urządzeniach rastrowych, przy selektywnym usuwaniu części obrazu, często pozostają małe "dziury" o wielkości piksela, w miejscu gdzie usuwane linie przecinają się z widocznymi liniami. Interakcyjne działanie na segmentach, wymaga jednoznacznej identyfikacji segmentów za pomocą nazw i wykonywania czynności m.in. transformacji niezależnie dla każdego segmentu. I tak np. można: obrócić Seg1 o 30° w kierunku zgodnym z ruchem wskazówek zegara i narysować go czerwoną linią ciągłą; usunąć z obrazu Seg2; przesunąć Seg3 do punktu (100,–20) i narysować go przerywaną linią niebieską. Dla rysunków składających się z segmentów, procedura tworzenia i wyświetlania ma postać:

Procedura Wyświetlania 2

Dla każdego segmentu należy wykonać co następuje:

Jeżeli segment ma być widoczny, to
Wykonać Procedurę Wyświetlania 1
W przeciwnym przypadku
Kontynuować.

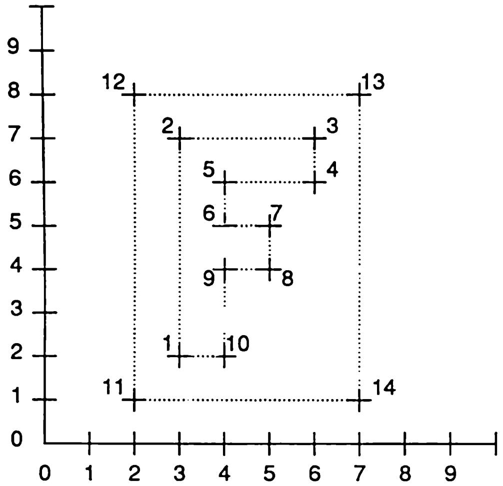
Warunkiem działania Procedury Wyświetlania 2 jest *struktura danych* przechowująca informacje o punktach, odcinkach, powierzchniach, atrybutach i transformacjach, w postaci łatwej do stosowania i zapamiętania.

11.6. Prosta graficzna struktura danych

Możliwe są różne formy struktury danych do interakcyjnego działania na segmentach; w książce zostanie opisana prosta struktura, w której są wykorzystywane tablice takie

jak dotychczas. Struktura ta wymaga pamięci dyskowej. Jest ona przydatna w większości systemów i języków programowania. W niektórych językach, jak Pascal czy C, są dopuszczalne bardziej złożone i silniejsze formy; prosta struktura może być łatwo zmodyfikowana i dostosowana do tych wymagań.

Dla każdego segmentu jest potrzebna lista współrzędnych świata dla punktów oraz lista numerów punktów, które mają być połączone odcinkami. Lista numerów jest tak zorganizowana, że punkty opisują brzegi każdej powierzchni, np. Segment 3 może składać się z litery F otoczonej prostokątem tak, jak pokazano na rys. 11.10.



Rys. 11.10. Współrzędne litery i prostokąt zawierający tę literę

W tym segmencie jest 14 punktów - dziesięć dla litery F i cztery dla prostokąta. Współrzędne punktów są następujące:

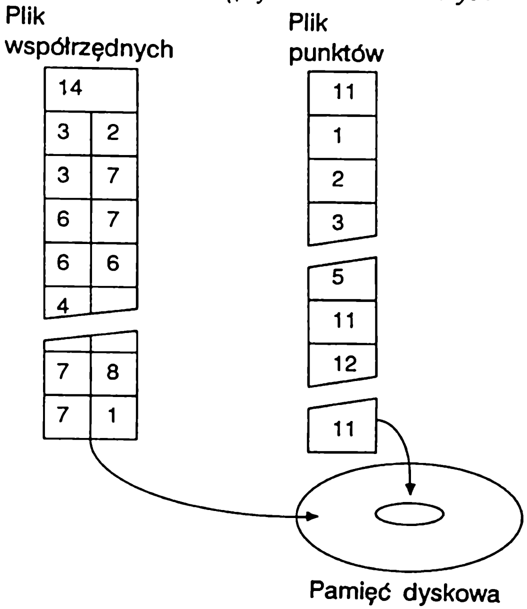
(3,2),(3,7),(6,7),(6,6),(4,6)
(4,5),(5,5),(5,4),(4,4),(4,2)
(2,1),(2,8),(7,8),(7,1)

Lista odcinków jest następująca:

11 /*liczba punktów dla litery F*/
1,2,3,4,5,6,7,8,9,10,1
5 /*liczba punktów dla prostokąta*/
11,12,13,14,11

Listy są pokazane na rys. 11.11 w postaci, w jakiej są przesłane na dysk; tzn. w postaci pliku (z nazwą) dla punktów i pliku (z nazwą) dla odcinków. Dodatkowo jest potrzebna *tablica segmentów*: tablica, która przechowuje informacje o segmentach i informuje system o szczegółach dotyczących segmentów. Tablica składa się z N wierszy, po jednym na segment i M kolumn po jednej na każdy rodzaj informacji.

W najprostszym systemie jest potrzebnych pięć kolumn tak, jak to pokazano na rys. 11.12. W pierszej kolumnie są przechowywane liczby punktów w segmencie;



Rys. 11.11. Zbiory dla obrazu z rysunku 11.10

Punkty	Odcinki	Widoczność	Atrybuty	Transformacje
20	22	1	1	1
4	5	1	1	1
14	18	1	3	1
12	12	0	2	0

Rys. 11.12. Przykładowa tablica segmentów

w drugiej kolumnie znajdują się liczby pozycji w listach odcinków; trzecia kolumna określa widoczność; czwarta informuje o atrybutach (takich jak rodzaj linii); piąta określa transformacje jakie mają być zastosowane do segmentu. Żeby uzyskać informacje o segmencie - powiedzmy o segmencie 3 - patrzymy na 3 wiersz tablicy i stwierdzamy, że powinniśmy się spodziewać 14 współrzędnych punktów i 18 elementów w liście odcinków. Ponieważ wskaźnik widoczności jest ustawiony na 1, to można otworzyć tablice na przyjęcie tych danych i odczytać je z dysku. Następnie można wykonać odpowiednie transformacje w odniesieniu do punktów i wyświetlić odcinki z określonymi atrybutami.

Dane o punktach i odcinkach są przechowywane oddzielnie w plikach dyskowych, którym nazwy są nadawane od ogólnej nazwy pracy i segmentu. Na przykład

dla punktów są nadane nazwy DOMPKT1, DOMPKT2 itd.; dla odcinków nazwy DOMODC1, DOMODC2 itd. Z opisu systemu operacyjnego można dowiedzieć się jaką postać mogą przyjmować te nazwy. Nasz system ma program tworzenia nazw, który bierze ogólną nazwę i łączy ją z numerem segmentu, tworząc w ten spsób jednoznaczny identyfikator. Z tablicą segmentu są związane trzy zmienne globalne:

- wolnyseg - numer następnego nieużywanego segmentu;
- bieżącyseg - numer bieżącego segmentu jeśli jest on wykorzystywany i 0 w przeciwnym przypadku;
- maksseg - dopuszczalna liczba segmentów.

Na to by móc łatwo i systematycznie dodawać, usuwać i zmieniać segmenty są potrzebne pewne podprogramy.

Jednym z nich jest podprogram Utwórz-segment (tabl. 11.8). Tablicaseg powinna być zadeklarowana na początku odwołania do Wymiar(Maxseg,5).

Tablica 11.8. Podprogram *Utwórz-segment*

```
Subroutine Utwórz-segment (NUM)
/* Podprogram otwiera segment o nazwie Num, ustawia system w stanie
gotowości do odebrania informacji o segmencie przez sprawdzenie, że wszystko
jest w porządku, ustawia domniemane opcje i otwiera pliki dyskowe */
        If Bieżącyseg > Num Then      /* już otwarty? */
           Print 'Segment już otwarty'
           Stop
        Endif
        If Num < 0 or Num > Maxseg Then   /* więcej miejsca? */
           Print 'Brak miejsca dla segmentów'
           Stop
        Endif
        If Num < Wolnyseg Then      /* już utworzony? */
           Print 'Już jest segment o takiej nazwie'
           Stop
        Endif
        /* Wszystko w porządku. Można kontynuować */
        Wolnyseg = Wolnyseg + 1
        Bieżącyseg = Num
        For J = 1 to 5    /* ustaw wartości domniemane */
           Tablseg(Num,J) = 0
        Next J
        Tablseg(Num,3) = 1    /*ustaw widoczność */
        Call Dyskop('open',Num)    /* otwórz pliki dyskowe */
End podprogramu Utwórz-segment
```

Dyskop jest podprogramem działania na plikach dyskowych. Jego szczegóły zależą od komputera; ogólna postać będzie zbliżona do tej podanej w tabl. 11.9.

Tablica 11.9. Podprogram *Dyskop*

```
Subroutine Dyskop(Łańcuchop,Num)
/* Podprogram otwiera lub zamyka pliki dyskowe i ustala ich nazwy */
        Call UstalNazwę(Pnazwa,Num;Pplik)    /*dla punktów */
        Call UstalNazwę(Lnazwa,Num;Lplik)    /*dla odcinków */
        If Łańcuchop = 'open' Then    /* otwórz plik */
           Open Pplik do połączenia jako *1
           Open Lplik do połączenia jako *2
        Else
           If Łańcuchop = 'close' Then    /* zamknij plik */
              Close
           Else
              Print 'Dyskop źle wywołane'    / * błąd */
              Stop
        Endif
End podprogramu Dyskop
```

Utwórz-segment nie pozwala na utworzenie segmentu jeżeli jakiś segment jest już otwarty. Stąd jest konieczne zamknięcie otwartego segmentu po skorzystaniu z jego danych. Robi to podprogram Zamknij-segment (tabl. 11.10).

Tablica 11.10. Podprogram *Zamknij-segment*

```
Subroutine Zamknij-segment
/* Podprogram zamyka bieżąco otwarty segment i wszystkie pliki dyskowe */
        If Bieżącyseg = 0 Then
           Print 'Nie ma otwartego segmentu'
           Stop
        Endif
        Call Dyskop('close',Bieżącyseg')
        Bieżącyseg = 0
End podprogramu Zamknij-segment
```

Nie trzeba określać, który segment należy zamknąć, ponieważ tylko jeden Bieżącyseg może być otwarty. Inny program jest potrzebny do umieszczania informacji w plikach dyskowych i jeszcze inny do jej odzyskania. Zadania te realizują podprogramy ZapiszDane (tabl. 11.11) i OdczytajDane (tabl. 11.12). Są one zależne od urządzeń i wymagają przystosowania do konkretnego systemu tak, żeby wykorzystać polecenia współpracy z dyskiem.

Tablica 11.11. Podprogram *ZapiszDane*

```
Subroutine ZapiszDane(Num,Punkty)
Integer Num
```

```
Real Punkty
/* Podprogram zapisuje dane w tablicy Punkty do otwartego pliku x1 */
Local Integer J
        Printx1,Num
        For J = 1 to Num
           Printx1, Punkty(J,1), Punkty(J,2)
        Next J
End podprogramu ZapiszDane
```

Tablica 11.12. Podprogram *OdczytajDane*

```
Subroutine OdczytajDane(Num,Punkty)
Integer Num
Real Punkty
/* Podprogram czytaj Punkty otwartego pliku x1 */
Local Integer J
        While Not EOF(1)   /* nie czytaj po końcu pliku */
           Inputx1,Num
           For J = 1 to Num
              Inputx1, Punkty(J,1), Punkty(J,2)
           Next J
        Endwhile
End podprogramu OdczytajDane
```

ZADANIA

11.1. Korzystając z instrukcji rysowania, dostępnych w systemie (zamiast MOVE i DRAW) napisz podprogram Wielokąt i sprawdź jego działanie dla różnych zbiorów Xtabl i Ytabl. Skorzystaj z programu podobnego do podanego w tabl. 11.4 używając podprogramów Ustaw-okno, Ustaw-wsPolePrezentacji i Wyznacz-skalę, opracowanych w zadaniach 10.3 i 10.5.

11.2. Napisz podprogram Kreskowanie z tabl. 11.5 i użyj go do wypełniania wielokątów różnymi wzorami. Spróbuj wykorzystać go do kreskowania wielokątów o różnych kształtach. W szczególności zakreskuj wielokąt opisany przez pary współrzędnych (10,0),(40,60),(70,0),(0,40),(80,40) i (10,0). Podprogram Kreskowanie wypełnia różne części tego samoprzecinającego się wielokąta, w swego rodzaju "standardowy" sposób. Czy to jest sposób jakiego można się było spodziewać? Jeżeli nie, to jaki inny sposób będzie logiczny?

11.3. Wykorzystując blisko siebie leżące linie poziome (na ekranie monitora w odstępie jednego piksela) pokaż, że podprogram Kreskowanie może być użyty do ciągłego wypełniania powierzchni. Jeżeli dostępny system ma swoją własne polecenie ciągłego wypełniania powierzchni, porównaj jego działanie z działaniem podprogramu Kreskowanie. Jeżeli nie jest dostępne ciągłe wypełnianie, opracuj inną wersję podprogramu Kreskowanie, powiedzmy Wypełnianie, w której nie

występują czasochłonne elementy obrotu. Korzystaj z tego podprogramu wówczas, gdy jest konieczne wypełnianie powierzchni.

11.4. W tekście pokazaliśmy metodę wypełniania powierzchni, w której wykorzystuje się fakt, że w niektórych systemach można stwierdzić, czy dany piksel jest rozjaśniony. Jeżeli dostępny system ma taką możliwość, napisz nowy podprogram wypełniania powierzchni, w którym wykorzysta się tę cechę systemu.

11.5. Jeżeli dostępny system ma wbudowaną funkcję wypełniania powierzchni, to sprawdź czy jest wykorzystywana metoda wypełniania wielokąta czy metoda malowania. Podaj zalety i wady obu metod. Jeżeli jest wykorzystywana metoda malowania, to jak można zapewnić, żeby punkt początkowy przekazywany systemowi przez program leżał wewnątrz rozważanego wielokąta? Co się stanie jeżeli zażąda się, żeby system wypełnił wielokąt, który nie jest domknięty? Czy podprogram Kreskowanie (lub zmodyfikowany) da w tym przypadku takie same rezultaty?

11.6. W niektórych, kolorowych systemach graficznych są ustalone zestawy kolorów, których nie można zmienić. W innych systemach na początku jest ustawiany domniemany zbiór kolorów; jest jednak możliwość jego zmiany. Jeżeli tak jest w twoim systemie, sprawdź czy zakłada on korzystanie z modelu RGB czy z jakiegoś innego. Nie jest łatwo pracować z RGB; wypraktykuj dobieranie kolorów dla różnych obiektów. W szczególności spróbuj ustawić różne odcienie brązowego i szarego. Pamiętaj o wskazówkach pomagających korzystać z systemu RGB podanych w tekście.

11.7. Spróbuj wykorzystać podprogram z odkłócaniem, metodą powierzchniową omówioną w tekście. Zoptymalizuj ten algorytm. Czy jest to metoda zadowalająca? Czy nasuwają się inne pomysły?

11.8. W grafice komputerowej ważne są sposoby działania na danych. Napisz i przetestuj podprogramy (podane w tekście) związane z segmentami i z pamiętaniem. Pamiętaj o korzystaniu z poleceń współpracy z dyskiem (dostępnych w systemie) w podprogramie z tabl. 11.9. Zwłaszcza pisząc podprogramy PiszDane i OdczytajDane korzystaj z poleceń umożliwiających zapisywanie na dysku i odczytywanie informacji.

11.9. Napisz i przetestuj podprogramy Doros i Punkt. Jeżeli w dostępnym systemie jest proste polecenie rysowania punktu, należy jego użyć zamiast podprogramu RysowanieOdcinka. Czy można podać sposoby modyfikowania podprogramu Doros do wypełniania koła lub rysowania elips?

12. Interakcja

Podane wcześniej podprogrogramy można wykorzystać do napisania programów, które będą odczytywały dane i tworzyły rysunki bez dalszej interwencji użytkownika. Jest to przykład nieinterakcyjnego korzystania z grafiki komputerowej. W wielu przypadkach użytkownik chce działać interakcyjnie, np. by realizować zadania graficzne opisane w rozdz. 2.; potrzebuje zatem podprogramów, które będą przyjmowały informacje z urządzeń wejściowych i odpowiednio je przetwarzały.

Są dwie podstawowe metody realizacji interakcji i do pewnego stopnia zależą one od własności urządzeń wejściowych i używanego komputera.

Pierwsza metoda jest określana jako **obsługa przerwań** (ang. *interrupt handling*). W tym przypadku sygnał wysyłany przez użytkownika za pomocą urządzenia wejściowego, zawiadamia procesor, że urządzenie wymaga obsługi i nastąpi wprowadzanie informacji. Sygnał ten powoduje przerwanie wykonywanego zadania i ustawienie systemu w stan gotowości do przyjęcia danych. Program musi zapamiętać bieżący stan programu, zanim zajmie się obsługą wejścia. Po zakończeniu obsługi przerwania, bieżący stan jest odtwarzany i przerwane zadanie jest kontynuowane.

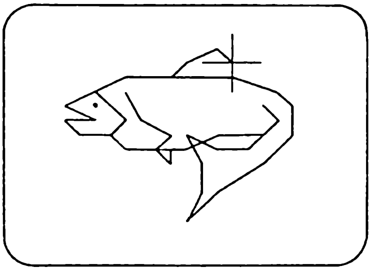
Ponieważ do systemu może być dołączone więcej niż jedno urządzenie wejściowe, w pewnych chwilach obsługa przerwań może być niewygodna, zatem musi istnieć możliwość programowego włączania i wyłączania urządzeń. Jest to raczej logiczne niż fizyczne włączanie i jest ono określane jako *aktywacja* bądź *dezaktywacja* urządzenia. Przerwania wywoływane zdarzeniami są wygodne w przypadku urządzeń wskazujących i przyciskowych.

Druga metoda jest znana jako **przeglądanie** (ang. *polling*). W tym przypadku program wchodzi w pętlę i oczekuje w niej na zainicjowaną przez użytkownika akcję w urządzeniu wejściowym, np. naciśnięcie przycisku lub klawisza. Będąc w pętli, system ciągle sprawdza czy oczekiwane zdarzenie wystąpiło. Jeżeli tak, to dane są przyjmowane i program rusza dalej. Jeżeli nie nastąpiło oczekiwane zdarzenie, program kontynuuje przeglądanie w pętli. Przeglądanie występuje np. w Basicu wówczas, gdy używa się instrukcji INPUT. Powoduje ona wejście w pętlę przeglądania i przejście systemu w stan oczekiwania na naciśnięcie klawisza Return.

W przypadku przeglądania sterowanego zdarzeniem, program będzie akceptował informację tylko wtedy, kiedy jest na to przygotowany. Wiele małych systemów może pracować tylko w tym trybie i nie mogą nic innego wykonywać wówczas, gdy są w pętli przeglądania. Załóżmy, że instrukcja graficzna ACCEPT jest poleceniem przeglądania i będą rozpatrywane zdarzenia odbierane w trybie przeglądania. Załóżmy również, że będąc w tym trybie system wyświetla na ekranie kursor (w postaci krzyżyka) tak, że jest możliwe zidentyfikowanie współrzędnych, które otrzyma instrukcja ACCEPT.

12.1. Lokatory

Do wskazania pozycji, można wybrać punkt na tabliczce lub na ekranie (jeżeli mamy pióro świetlne) i zapytać komputer o akceptację współrzędnych tego punktu. Do zainicjowania tego procesu należy nacisnąć przycisk, a program musi pozostawać w podprogramie oczekiwania na współrzędne. Wprowadzając nowe punkty, jeden po drugim, można rysować linie (rys. 12.1). Zakończenie serii odcinków linii, można zgłosić przez wskazanie specjalnie przeznaczonej do tego celu części powierzchni do wpro-



Rys. 12.1. Kursor wskazujący bieżącą pozycję pióra

wadzania, albo nacisnąć przycisk "koniec tej czynności", jeżeli lokator ma więcej niż jeden przycisk. Wprowadzane współrzędne są podawane we współrzędnych urządzenia i przed korzystaniem z nich, musimy zamienić je na współrzędne świata. W tablicy 12.1 jest pokazany podprogram przyjmowania danych i rysowania metodą wskazywania kolejnych punktów rysunku.

Tablica 12.1. Podprogram *TworzenieLinii*

```
Subroutine TworzenieLinii(N,Otablica)
Integer N
Real Otablica
/* Podprogram akcepuje współrzędne co najwyżej Maxnum-1 punktów z lokatora
i wyświetla łączące je odcinki. N par współrzędnych jest pamiętanych w Otablicy.
Gdy wskaźnik F lokatora jest równy 2 lub Y1 < YD (najniższa część powierzchni
do rysowania) podprogram kończy pracę i dalsze współrzędne są pomijane.
Wartość YD powinna być zdefiniowana globalnie poza podprogramem */
```

```
Global Integer YD
Local Integer   Maxnum, J ,F, X1, Y1
Local Real Poınoc, X, Y
        Maxnum = 100
        N = 0
        Dim Pomoc(Maxnum,2)   /* tablica pomocnicza */
        For J = 1 to Maxnum
          ACCEPT (F,X1,Y1)
          If F = 2 or Y1 < YD Then
            J = Maxnum + 1   /* gotowe do zakończenia */
          Else
            N = N + 1   /* liczba par współrzędnych */
            Call Zamiana(X1;X)   /* przeliczenie do współrzędnych świata */
            Call Zamiana(Y1;Y)
            Pomoc(J,1) = X
            Pomoc(J,2) = Y
            If J = 1 Then   /* rysowanie */
              MOVE (X1,Y1)
            Else
              DRAW (X1,Y1)
            Endif
          Endif
        Next J
/* Ładuj współrzędne do Otablica */
        Dim Otablica (N,2)
        For J = 1 to N
          Otablica(J,1) = Pomoc(J,1)
          Otablica(J,2) = Pomoc(J,2)
        Next J
End podprogramu TworzenieLinii
```

12.2. Wskazywanie i wybieranie

W trybie interakcyjnym często zachodzi potrzeba wybrania obiektu na ekranie przez wskazanie go za pomocą kursora i urządzenia wskazującego. Obiektami tymi mogą być elementy menu lub części rysunku takie jak punkty, linie lub całe segmenty.

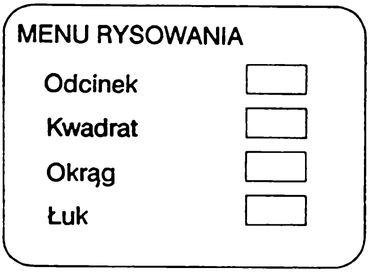
Wybór z menu

Chcąc dokonać wyboru z menu, które może być zbudowane z tablicy prostokątów tak, jak pokazano na rys. 12.2, wskazujemy punkt (PX,PY) należący do odpowiedniego prostokąta. Następnie porównujemy współrzędne wybranego punktu z brzegami każ-

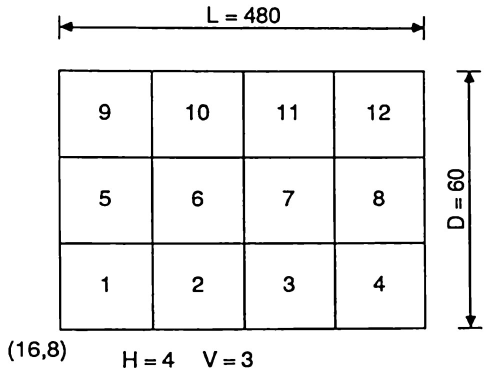
dego prostokąta. Można to zrobić, biorąc pod uwagę współrzędne (Xminp, Yminp, Xmaxp, Ymaxp) każdego prostokąta i sprawdzić czy:

Xminp < PX i Xmaxp > PX i Yminp < PY i Ymaxp > PY

Jeżeli wynik jest prawdą, to został znaleziony wskazany prostokąt. Jednak takie szukanie jest stosunkowo powolne i nie wykorzystuje uporządkowania i jednakowych wymiarów wszystkich prostokątów. Jeżeli przyjmiemy strukturę menu taką jak przedstawiono na rys. 12.3, to podprogram WybórZMenu (tabl. 12.2) znajdzie prostokąt, w którym leży punkt (PX,PY). Po znalezieniu obiektu w menu, możemy przejść do rysowania, wykorzystując program z wywołaniem podprogramów WybórZMenu i TworzenieLinii (tabl. 12.3).



Rys. 12.2. Typowe menu ekranowe



Rys. 12.3. Struktura menu

Tablica 12.2. Podprogram *WybórZMenu*

Subroutine WybórZMenu(PX,PY;IDnum)
Integer PX, PY, IDnum
/* Podprogram daje w wyniku liczbę IDnum wskazującą, który prostokąt z menu został wybrany za pomocą współrzędnych (PX,PY). Menu jest zdefiniowane globalnie przez wymiary L x D; liczbę prostokątów w poziomie H i w pionie V; współrzędne dolnego lewego rogu (XO,YO). Jeżeli IDnum = 0, to (PX,PY) nie należy do przestrzeni menu. Wszystkie wymiary są we współrzędnych urządzenia */

```
Local Integer NX, NY
Local Real SX, SY
        SX = L/H   /* szerokość prostokąta menu */
        SY + D/V   /* wysokość prostokąta menu */
        NX = Int ((PX - X0)/SX) + 1
        NY = Int ((PY - Y0)/SY) + 1
        IDnum = (NY - 1) * H + NX
        If IDnum > H * V or IDnum < 1 Then
           IDnum = 0
        Endif
End podprogramu WybórZMenu
```

Tablica 12.3.

```
/* Fragment programu rysowania linii za pomocą lokatora */
        L = 480  /* parametry menu we współrzędnych urządzenia */
        D = 60
        X0 = 16
        Y0 = 8
        H = 4    /* menu ma 12 prostokątów */
        V = 3
        Wskaźnik = 0
        While Wskaźnik = 0 Do    /* dopóki nie jest ustawiony wskaźnik końca */
           ACCEPT (F, PX, PY)
           Call WybórZMenu(PX,PY;IDN)
              Case IDN of:
              1: Call TworzenieLinii(Num,OA) /* linie łączące */
              2:                 /* następna czynność */
              .
              .
              .
              12: Wskaźnik = 1    /* koniec */
              End of Case
        End while
/* Koniec fragmentu programu */
```

Wskazywanie punktów

Do wskazania punktu, który jest już wyświetlony na ekranie i jest zapisany w tablicy, należy pobrać współrzędne tego punktu (ACCEPT) i porównać je ze współrzędnymi zapisanymi w tablicy. Byłoby nierozsądne spodziewać się, że można dokładnie wskazać właściwe współrzędne, wybierając punkt za pomocą urządzenia wskazującego. Można jednak z grubsza wskazać pozycję i na podstawie tego wskazania system znajdzie dla nas dokładnie ten punkt, o który nam chodzi. Żeby to uzyskać, należy zdefiniować

obszar wokół wskazywanych współrzędnych i szukać punktu, który w nim leży. Jest to operacja odwrotna do zadania wskazywania w menu, w której punkt jest dany, a celem jest znalezienie obszaru, do którego ten punkt należy.

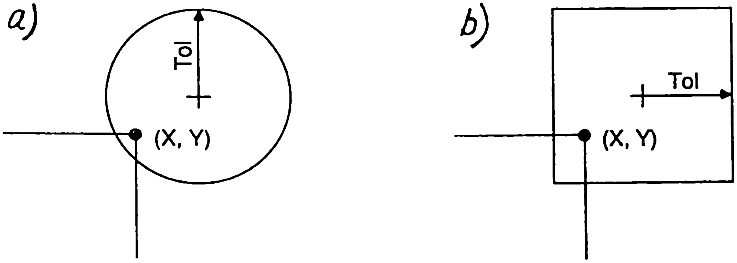
Kształt obszaru otaczającego wskazywany punkt nie jest obojętny. Najlepszym kształtem jest koło, ponieważ jego brzeg leży w stałej odległości od wskazywanego punktu (rys. 12.4a). Przy założeniu, że promień koła wynosi Tol, można sprawdzić czy punkt (X,Y) leży wewnątrz koła. Należy wykonać sprawdzenie, czy

$$(PX - X)^2 + (PY - Y)^2 < Tol^2$$

dla każdego punktu w tablicy. Szybsza metoda polega na zdefiniowaniu kwadratu o boku 2*Tol (rys. 12.4b). Wtedy należy tylko sprawdzić, czy

$$Abs(PX - X) + Abs(PY - Y) < Tol$$

Jednak kwadraty nie są intuicyjnie łatwe w stosowaniu, zwłaszcza gdy nie chcemy wyświetlać obszaru w czasie wskazywania. Wyświetlanie można stosować wygodnie tylko wtedy, kiedy system pozwala na definiowanie własnego kursora.



Rys. 12.4. a) koło tolerancji o środku w (PX, PY) zawiera punkt (X, Y); b) prostokąt tolerancji o środku w (PX, PY) zawiera punkt (X, Y)

Niezależnie od tego jaki kształt otoczenia przyjmujemy, początkowo (przez domniemanie) musimy wybrać jakąś rozsądną wartość Tol. W miarę jak wzrasta liczba punktów występujących w naszym rysunku stwierdzimy, że jest coraz trudniej wskazać pojedynczy punkt przy początkowej wartości tolerancji. Wystarcza wówczas odpowiednio zmniejszyć wartość Tol. Na ogół należy zmniejszać (lub zwiększać) tę wartość dwukrotnie. Napisanie programów wskazywania punktów i zmiany tolerancji zostawiamy jako zadanie dla Czytelnika.

12.3. Wskazywanie odcinków

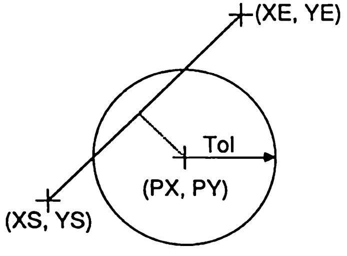
Wskazywanie odcinków wymaga większej ilości obliczeń niż wskazywanie punktów. Jeżeli mamy zbiór odcinków, których końce są zapisane w tablicy, to należy przejrzeć tablicę i sprawdzić, czy odległość między wybranym punktem (PX,PY) a dowolnym

odcinkiem jest mniejsza od pewnej zadanej odległości Tol, która definiuje obszar tolerancji wskazania (rys. 12.5). Jeżeli obszar ten jest kołem, a końce odcinka są oznaczone jako (XS,YS) i (XE,YE), to obliczenia, które należy wykonać są następujące:

$U = ((PX - XS) * (YE - YS) - (PY - YS) * (XE - XS))^2$

$L = (XE - XS)^2 + (YE - YS)^2$

Odcinek jest wybrany, jeżeli $U/L < Tol^2$.



Rys. 12.5. Wykorzystanie koła tolerancji do znalezienia odcinka przez obliczenie odległości od (PX,PY)

W celu uniknięcia tych dość złożonych obliczeń dla odcinków, które nie będą brane pod uwagę, należy je w pierwszej kolejności wyeliminować. Są to odcinki, dla których obszar tolerancji leży poza punktami końcowymi. Można więc natychmiast odrzucić odcinek, jeżeli któryś z następujących warunków jest spełniony:

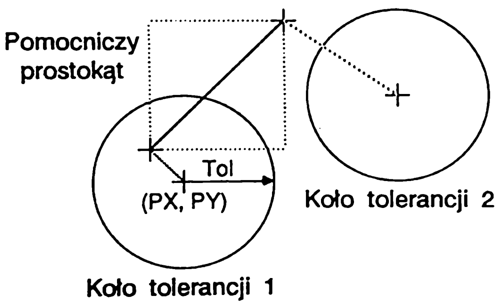
$PX < Min(XS,XE) - Tol$

$PX > Max(XS,XE) + Tol$

$PY < Min(YS,YE) - Tol$

$PY > Max(YS,YE) + Tol$

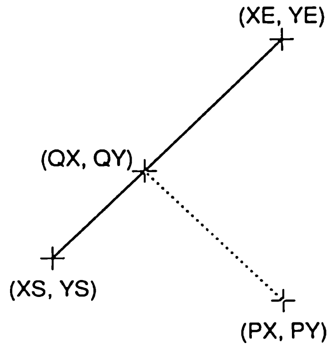
Wykonane obliczenia odpowiadają sprawdzeniu, czy punkt pomocniczy wskazuje na prostokąt, którego przekątną jest badany odcinek (rys. 12.6). Jeżeli wybrany punkt nie



Rys. 12.6. Wskazywanie odcinka z wykorzystaniem pomocniczego prostokąta (odcinek może być wzięty pod uwagę dla koła tolerancji 1, natomiast dla koła tolerancji 2 - nie

należy do prostokąta, to odcinek nie może być tym poszukiwanym. Jeżeli jednak punkt leży wewnątrz prostokąta, to dalej należy sprawdzać, czy odcinek jest tym poszukiwanym; mogą bowiem istnieć inne odcinki, które mają taką samą właściwość.

Czasami jest konieczne zidentyfikowanie odcinka i punktu do niego należącego, który leży najbliżej wskazywanego punktu (rys. 12.7).



Rys. 12.7. Punkt (QX,QY) leży najbliżej punktu (PX,PY) spośród wszystkich punktów odcinka (XS,YS), (XE,YE)

Można to zrobić za pomocą następującej procedury:

- zidentyfikować odcinek,
- znaleźć wartość T dla punktu Q leżącego na odcinku, korzystając z równania

$$T = ((XE - PX) * (XE - XS) + (YE - PY) * (YE - YS))/L$$

gdzie L jest zdefiniowane jak wyżej,

- do zlokalizowania najbliższego punktu (QX,QY) wykorzystać wartość T w parametrycznym równaniu odcinka.

Metoda ta jest wykorzystana w podprogramie przedstawionym w tabl. 12.4.

Tablica 12.4. Podprogram *PunktNaOdcinku*

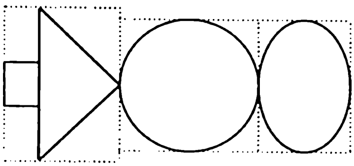
```
Subroutine PunktNaOdcinku(N,Ltab,Dtol,PXD,PYD;Num,WX,WY)
Integer N, Num, Dtol, PXD, PYD
Real Ltab, WX, WY
/* Podprogram przeszukuje Ltab zawierającą L zbiorów współrzędnych końców
odcinków w celu znalezienia pierwszego odcinka, który znajduje się w odległości
Dtol od wskazanego punktu (PXD,PYD). W wyniku uzyskuje się numer Num wska-
zujący pozycję (w tabeli) punktu początkowego odcinka i współrzędne świata
(WX,WY) punktu na odcinku, najbliższego do punktu (PXD,PYD). Jeżeli Num = 0,
to nie ma odcinka spełniającego warunek i WX, WY są niezdefiniowane. PXD, PYD
i Dtol są podane we współrzędnych urządzenia. Ltab zawiera dane we współ-
rzędnych świata */
Local Integer J
Local Real PX, PY, Tol, XS, XE, XD, XL, YS, YE, YD, YL, U, UU, L, T
        Call Zamiana (PXD; PX)   /* przeliczenie do współrzędnych świata */
```

```
        Call Zamiana (PYD; PY)
        Call Zamiana (Dtol; Tol)
        Num = 0
        For J = 1 to N - 1
           XS = Ltab(J,1)   /* początek odcinka */
           YS = Ltab(J,2)
           XE = Ltab(J + 1,1)   /* koniec odcinka */
           YE = Ltab(J + 1,2)
/* Tu byłoby wstępne sprawdzenie z pomocniczym prostokątem */
           XD = XE - XS
           YD = YE - YS
           XL = PX - XS
           YL = PY - YS
           U = XL * YD - YL * YD
           UU = U * U
           L = XD * XD + YD * YD
/* Sprawdzenie tolerancji */
           If UU/L  Tol * Tol Then
              T = (XL * XD + YL * YD)/L
              If T > = 0 and T < = 1 Then
                 WX = XS + T * XD
                 WY = YS + T * YD
                 Num = J
                 J = N   /* wyjście: znaleziony odcinek */
              Endif
           Endif
        Next J
 End podprogramu PunktNaOdcinku
```

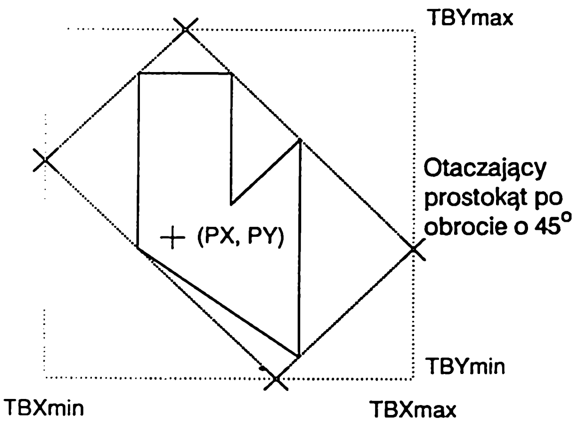
Wskazywanie segmentów

Segment można wskazać przez zidentyfikowanie jednego z jego odcinków znalezionych metodą przeszukiwania, tak jak w podprogramie PunktNaOdcinku. Dla dużej liczby segmentów i odcinków, może to być powolny proces. Można go przyspieszyć, jeżeli w tablicy segmentów są przechowywane również współrzędne (Bxmin,Bxmax, Bymin,Bymax) prostokątów, które otaczają każdy segment (rys. 12.8). Korzystając z tych prostokątów, można robić wstępne sprawdzenie, czy segment może być brany pod uwagę, a dopiero potem sprawdzać jego części składowe. Przed wyświetleniem na ekranie segment może być poddany obrotowi, przesunięciu lub skalowaniu. W tym przypadku współrzędne otaczającego prostokąta powinny być również poddane takim działaniom, zanim rozpocznie się poszukiwanie.

Do sprawdzenia, czy badany punkt (PX,PY) należy do przekształconego prostokąta, korzysta się ze współrzędnych po transformacjach (TBXmin, TBXmax, TBYmin, TBYmax) (rys. 12.9).



Rys. 12.8. Prostokąty otaczające segmenty



Rys. 12.9. Współrzędne otaczającego prostokąta po dokonaniu obrotu

Segment jest brany pod uwagę jeżeli
PX > TBXmin i PK < TBXmax i PY > TBYmin i PY < TBYmax

Ponieważ tylko widoczne segmenty mogą być wskazywane, należy sprawdzić atrybut widzialności zanim zostanie wykonany test z prostokątem otaczającym segment. Następnie, do pozostałych elementów segmentów, wykorzystuje się podprogram PunktNaOdcinku. Do znalezienia wartości najmniejszej i największej w tablicy, wykorzystuje się program pokazany w tabl. 12.5.

Tablica 12.5. Podprogram *Ekstrema*

```
Subroutine Ekstrema(N,OA;X0,Y0,X9,Y9)
Integer N
Real OA, X0, X9, Y0, Y9
/* Podprogram znajduje wartości minimalne X0, Y0 i maksymalne X9, Y9 w tablicy
OA zawierającej N punktów */
Local Integer J
     X0 = OA(1,1)   /* załóż, że pierwszy element jest min i max */
```

```
   Y0 = OA(1,2)
   X9 = OA(1,1)
   Y9 = OA(1,2)
   For J = 2 to N
      If OA(J,1) < X0 Then
         X0 = OA(J,1)    /* nowe min X */
      Else
         If OA(J,1) > X9 Then
            X9 = OA(J,1)    /* nowe max X */
         Endif
      Endif
   If OA(J,2) < Y0 Then
         Y0 = OA(J,2)    /* nowe min Y */
   Else
         If OA(J,2) > Y9 Then
            Y9 = OA(J,2)    /* nowe max Y */
         Endif
      Endif
   Next J
End podprogramu Ekstrema
```

ZADANIA

12.1. Jeżeli dostępny system jest wyposażony w mysz, wolant, tabliczkę lub inny lokator, to wykorzystaj go razem z podprogramem TworzenieLinii do opracowania rysunków i zapamiętania punktów. Należy zastąpić ACCEPT, MOVE i DRAW odpowiednimi poleceniami dostępnymi w systemie.

12.2. Użyj podprogramu WybórZMenu i dostępnego urządzenia wejściowego do wybierania obiektów z menu dziesięcio- lub więcej elementowego, dostępnego na ekranie lub tabliczce. Sprawdź działanie podprogramu WybórZMenu wskazując obszary leżące wewnątrz i na zewnątrz menu.

12.3. Napisz i przetestuj podprogramy do wskazywania punków i zmieniania tolerancji, wykorzystując metody omówione w tekście. W szczególności sprawdź, czy istotnie zwiksza się szybkość interakcji przy używaniu prostokąta zamiast koła tolerancji (rys. 12.4a). Jeżeli nawet zwiększa się szybkość, to czy korzystanie z prostokąta tolerancji jest tak samo wygodne jak korzystanie z koła? Jeżeli w dostępnym systemie można zdefiniować kursor, wskaż punkty przy wyświetlanym kole, bądź prostokącie tolerancji. Czy poprawia to jakość interakcji w porównaniu ze wskazywaniem za pomocą zwykłego kursora? Pomyśl o innych ulepszeniach.

12.4. Napisz podprogram PunktNaOdcinku i wykorzystaj go do wykrywania odcinków i najbliższych punktów. Po pełnym przetestowaniu podprogramu, użyj go razem z podprogramem wskazywania punktu z zadania 12.3 do wskazywania segmentów. Do sprawdzenia, czy można uzyskać znaczące oszczędności czasowe spró-

buj: pełnego przeszukiwania danych dla złożonego rysunku i metodę otaczającego prostokąta podaną w tekście.

12.5. Identyfikacja wskazanego segmentu, który był poddany kilku transformacjom nie jest zadaniem trywialnym. W tekście zasugerowaliśmy sposób realizacji tej operacji. Czy można opracować bardziej efektywne metody?

12.6. Nawet przy małych kołach tolerancji może się okazać, że więcej niż jeden punkt może być brany pod uwagę. Który punkt powinien być zidentyfikowany: pierwszy na liście punktów, czy ostatni znaleziony? Czy możesz wydrukować komunikat mówiący użytkownikowi, że wskazanie jest dwuznaczne? Przemyśl różne warianty i podaj ich zalety i wady. W szczególności podaj metodę wskazywania użytkownikowi, że nie ma punktów wewnątrz koła tolerancji. Należy pamiętać, że w większości komputerów jest możliwe generowanie dźwięków i można użyć ich ostrzegania, zamiast (lub obok) metod wzrokowych.

13. Modelowanie obiektów trójwymiarowych

Obiekty trójwymiarowe mogą być opisane na wiele sposobów; niektóre z nich są takie same jak dla obiektów dwuwymiarowych. Na przykład można korzystać z punktów i odcinków do reprezentacji wierzchołków i krawędzi wielościanu tak, jak pokazano na rys. 13.1. Przedstawiony sześcian ma osiem wierzchołków umieszczonych w następujących pozycjach:

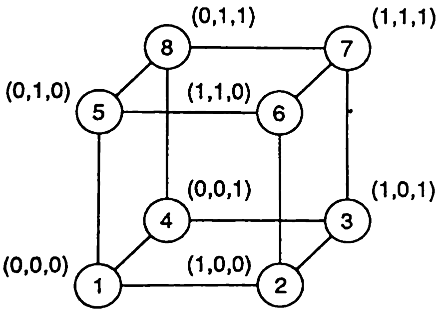
(0,0,0),(1,0,0),(1,0,1),(0,0,1)
(0,1,0),(1,1,0),(1,1,1),(0,1,1)

Trójki współrzędnych można ponumerować od 1 do 8 i utworzyć listę punktów końcowych odcinków zawierającą dwanaście krawędzi:

(1,2),(2,3),(3,4),(4,1),(1,5),(5,6)
(6,2),(6,7),(7,3),(7,8),(8,4),(8,5)

Jeżeli czytelnik ma trudności z wyobrażeniem sobie tych punktów i odcinków, to warto wziąć małe pudełko i opisać jego wierzchołki współrzędnymi i liczbami.

Podany opis jest wystarczający dla prostych przypadków, dla innych jest zbyt skromny. W szczególności zawiera on mało informacji o powierzchniach sześcianu, potrzebnych np. wtedy, kiedy będą wymagały pokolorowania. Jeden ze sposobów uzu-



Rys. 13.1. Sześcian ze wskazanymi współrzędnymi wierzchołków

pełnienia brakujących informacji polega na zrobieniu uporządkowanej listy współrzędnych wierzchołków, związanych z każdą ścianą boczną. Należy zrobić ją w konsekwentny sposób; powiedzmy zgodnie z ruchem wskazówek zegara, jeśli patrzymy na ścianę sześcianu z zewnątrz. To systematyczne podejście służy upewnieniu się, że wszystkie punkty zostały poprawnie uwzględnione i określeniu kierunku, w którym jest skierowana płaszczyzna po obrocie

```
4 /*ściana południowa*/
0,0,0,0,1,0,1,1,0,1,0,0
4 /*ściana wschodnia*/
1,0,0,1,1,0,1,1,1,1,0,1
4 /*ściana północna*/
1,0,1,1,1,1,0,1,1,0,0,1
4 /*ściana zachodnia*/
0,0,1,0,1,1,0,1,0,0,0,0
4 /*góra*/
0,1,0,0,1,1,1,1,1,1,1,0
4 /*dół*/
0,0,0,1,0,0,1,0,1,0,0,1
0 /*koniec*/
```

W tym opisie każda współrzędna jest użyta trzykrotnie, nie jest on więc oszczędny, biorąc pod uwagę zajęty obszar pamięci, zwłaszcza gdy opisuje się większą liczbę bardziej złożonych obiektów. Co ważniejsze, każdy punkt musi być trzykrotnie poddany transformacji przez każdy program. Opis ten jest prosty do zrozumienia i utworzenia, ale podobnie jak w układzie współrzędnych dwuwymiarowych, wygodniej jest przechowywać punkty i krawędzie w oddzielnych listach:

```
/*Lista wierzchołków sześcianu*/
8 /*lista wierzchołków*/
0,0,0,1,0,0,1,0,1,0,0,1
0,1,0,1,1,0,1,1,1,0,1,1
/*lista ścian - uporządkowanie zgodne z ruchem wskazówek zegara*/
4 /*ściana południowa*/
1,5,6,2
4 /*ściana wschodnia*/
2,6,7,3
4 /*ściana północna*/
3,7,8,4
4 /*ściana zachodnia*/
4,8,5,1
4 /*góra*/
5,8,7,6
4 /*dół*/
1,2,3,4
0 /*koniec*/
```

Korzystając z tych dwóch list można dokonać transformacji dla każdego punktu tylko raz, bez utraty informacji o sposobie połączenia krawędzi i ścian.

Opisując wypukłe wielościany, warto pamiętać, że liczba wierzchołków, liczba krawędzi i liczba ścian są ze sobą związane prostą regułą, znaną jako *twierdzenie Eulera* o wielościanach. Brzmi ono następująco:

Liczba wierzchołków + liczba ścian – liczba krawędzi = 2

Stosując twierdzenie należy sprawdzić, czy mamy dostateczną ilość informacji do opisania sześcianu. Stwierdzamy, że liczba wierzchołków w liście jest równa 8, liczba ścian jest równa 6 i wobec tego potrzebujemy $8+6-2=12$ krawędzi. Faktycznie w liście ścian znajdują się 24 krawędzie, ale każda jest podana dwukrotnie, w dwóch przeciwnych kierunkach. Stąd liczby podanych elementów są poprawne.

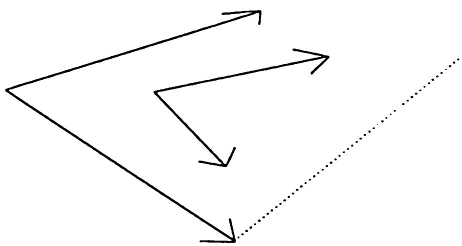
13.1. Płaszczyzny

Wiadomo, że dowolne dwa różne punkty jednoznacznie definiują odcinek i współrzędne punktów na odcinku są związane zależnością:

$$AX + BY + C = 0$$

W przypadku układu trójwymiarowego można zauważyć, że dowolne dwa, przecinające się odcinki jednoznacznie definiują płaszczyznę (rys. 13.2) i wszystkie punkty na płaszczyźnie są związane zależnością:

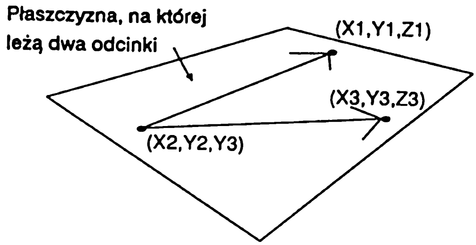
$$AX + BY + CZ + D = 0$$



Rys. 13.2. Dowolne dwa odcinki wychodzące z tego samego punktu definiują płaszczyznę

Przy założeniu, że dwa odcinki zaczynają się w tym samym punkcie początkowym (X2,Y2,Z2) i ich punkty końcowe mają odpowiednio współrzędne (X1,Y1,Z1) i (X3,Y3,Z3) jak pokazano na rys. 13.3, można wyznaczyć współczynniki A,B,C i D z zależności

$$XA = X1 + X2$$
$$XB = X2 + X3$$
$$XC = X3 + X1$$
$$YA = Y1 + Y2$$



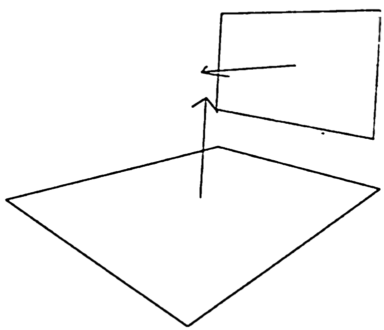
Rys. 13.3. Dwa wektory na płaszczyźnie

YB = Y2 + Y3
YC = Y3 + Y1
ZA = Z1 + Z2
ZB = Z2 + Z3
ZC = Z3 + Z1
A = (Y1 - Y2) * ZA + (Y2 - Y3) * ZB + (Y3 - Y1) * ZC
B = (Z1 - Z2) * XA + (Z2 - Z3) * XB + (Z3 - Z1) * XC
C = (X1 - X2) * YA + (X2 - X3) * YB + (X3 - X1) * YC
D = -(A*X1 + B*Y1 + C*Z1)

W zależnościach występuje symetria; dowolne trzy punkty określają płaszczyznę pod warunkiem, że nie są kolinearne tzn. nie leżą na jednej prostej.

13.2. Normalne

W grafice komputerowej, często musi być znany kierunek, w którym są zwrócone ściany boczne obiektu. Można sobie wyobrazić linię prostopadłą do powierzchni, wy-



Rys. 13.4. Normalne do powierzchni

chodzącą z jednej jej strony tak, jak przedstawiono na rys. 13.4. Linia ta jest określana jako *normalna do powierzchni* i kierunek przez nią wskazywany definiuje orientację ściany. Płaszczyzna o współczynnikach A,B,C i D ma normalną wychodzącą z punktu (XP,YP,ZP) skierowaną do punktu (A+XP, B+YP, C+ZP). Zatem płaszczyzna opisana równaniem

$$3 * X + 2 * Y + 6 * Z - 23 = 0$$

ma normalną poprowadzoną od punktu (3,4,1) na płaszczyźnie do punktu (6,6,7).

Są przypadki, w których jest wymagana normalna o długości równej 1. Uzyskuje się ją, dzieląc każdy współczynnik A, B i C przez pierwiastek kwadratowy sumy kwadratów A, B i C. Zatem, jeżeli

$$L = \sqrt{A^2+B^2+C^2}$$

to jednostkowa normalna wychodząca z punktu (XP,YP,ZP) jest skierowana do punktu

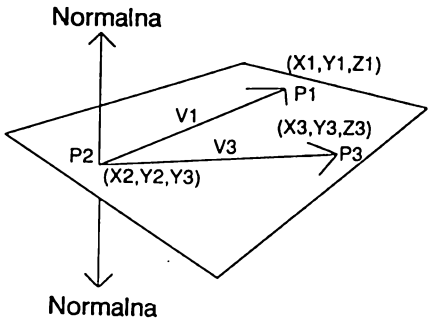
$$(A/L + XP, B/L + YP, C/L + ZP)$$

Na przykład jednostkowa normalna wychodząca z punktu (3,4,1) na płaszczyźnie o współczynnikach (3,2,6,–23) jest poprowadzona od (3,4,1) do punktu (24/7,30/7,13/7). Przejście normalnej na zewnątrz od przodu płaszczyzny lub do wewnątrz od tyłu płaszczyzny zależy od sposobu opisu płaszczyzny, który jest ukierunkowany kolejnością określenia trzech punktów użytych do wyznaczenia współczynników. Jeżeli punkty te są podawane w kolejności zgodnej z ruchem wskazówek zegara, to normalna wskazuje jeden kierunek, jeżeli natomiast są podawane w kolejności przeciwnej do ruchu wskazówek zegara, to normalna wskazuje przeciwny kierunek. Można stwierdzić to definiując płaszczyznę za pomocą trzech punktów (3,1,4), (0,0,0) i (1,4,3). Wstawiając współrzędne punktów w podanym porządku do równania płaszczyzny, otrzymamy współczynniki (13,5,–11,0) oraz normalną w punkcie (0,0,0) skierowaną do punktu (13,5,–11). Wstawiając współrzędne punktów w odwrotnej kolejności otrzymamy współczynniki (–13,–5,11,0) i normalną w punkcie (0,0,0) skierowaną do punktu (–13,–5,11). Kierunki tych normalnych są przeciwne. Korzystając z układu współrzędnych i podanego równania płaszczyzny otrzymamy obiekt, którego ściany są uporządkowane w kierunku zgodnym z ruchem wskazówek zegara, a normalne ścian są skierowane na zewnątrz. W algorytmach przytoczonych w książce przyjęto, że opis płaszczyzn jest w porządku zgodnym z ruchem wskazówek zegara.

13.3. Normalne i widoczność

Istnieje inna metoda obliczania normalnej do powierzchni, w której nie trzeba wyznaczać współczynników płaszczyzny. Wykorzystuje się w niej iloczyn wektorowy (Dodatek 1). Dla danych trzech punktów P1(X1,Y1,Z1), P2(X2,Y2,Z2) i P3(X3,Y3,Z3) leżących na płaszczyźnie można narysować wektory poprowadzone od P2 do P1 i od P2 do P3 tak, jak pokazano na rys. 13.5.

Iloczyn wektorowy dwóch wektorów V1 i V3 daje w wyniku nowy wektor, który jest normalny do powierzchni w punkcie P2. Kolejność, w jakiej są podane



Rys. 13.5. Iloczyn wektorowy

punkty wpływa na kierunek normalnej. Iloczyn wektorowy V1 × V3 daje normalną w punkcie P2 skierowaną w jedną stronę, a iloczyn V3 × V1 daje normalną w punkcie P2 skierowaną w przeciwną stronę. Na przykład iloczyn wektorowy (3,1,4) × (1,4,3) daje normalną w punkcie (0,0,0) skierowaną do punktu o współrzędnych (–13,–5,11), natomiast iloczyn wektorowy (1,4,3) × (3,1,4) daje normalną w punkcie (0,0,0) skierowaną do punktu o współrzędnych (13,5,–11). Oznacza to, że jeżeli widzimy punkty P1, P2 i P3 na płaszczyźnie uporządkowane zgodnie z ruchem wskazówek zegara, to iloczyn wektorowy V3 × V1 da w wyniku normalną skierowaną do nas.

Kierunek normalnej można wykorzystać do określenia, czy po dokonaniu obrotu, ściana obiektu będzie widoczna. Oczywiście w przypadku brył będą widoczne tylko te ściany, których normalne są skierowane do nas. Kilka chwil eksperymentu z ołówkiem stojącym na książce przekona, że powierzchnia jest skierowana od nas, jeżeli kąt między prostą biegnącą od punktu obserwacji, a normalną jest większy niż 90° lub mniejszy niż -90°. Wyznaczając kąt między prostą widzenia a normalną do powierzchni w punkcie przecięcia z prostą widzenia, można sprawdzać, czy określona płaszczyzna będzie widoczna. Na szczęście w rozwiązaniu tego zadania pomaga nam algebra wektorów. Jak pokazano w Dodatku 1, do znalezienia kąta między dwoma wektorami można użyć iloczynu skalarnego. Dokładniej, iloczyn skalarny wyznacza cosinus kąta między dwoma wektorami o długości 1 (jednostkowej). Cosinusy kątów przyjmują wartości z przedziału (–1, + 1). Cosinusy kątów z przedziału $(-90^{\circ},90^{\circ})$ przyjmują wartości z przedziału (0,1). Tak więc, żeby obliczyć wartość kąta, musimy znaleźć jednostkowy wektor normalny do powierzchni i obliczyć jego iloczyn skalarny z wektorem jednostkowym skierowanym do punktu obserwacji. Jeżeli wartość tego iloczynu jest mniejsza od 0, to płaszczyzna jest skierowana od nas. W tablicy 13.1 jest podany podprogram obliczania wektora jednostkowego o zadanych punktach: początkowym i końcowym.

Tablica 13.1. Podprogram *WektorJednostkowy*

```
Subroutine WektorJednostkowy(XS,YS,ZS,XE,YE,ZE;XU,YU,ZU)
Real XS, XE, XU, YS, YE, YU, ZS, ZE, ZU
```

```
/* Podprogram oblicza wktor jednostkowy skierowany od punktu (XS,YS,ZS)
do punktu (XE,YE,ZE) */
Local Real L1, L2, L3, UL
        L1 = XE - XS
        L2 = YE - YS
        L3 = ZE - ZS
        UL = Sqr(L1 * L1 + L2 * L2 + L3 * L3)
        XU = L1/UL + XS
        YU = L2/UL + YS
        ZU = L3/UL + ZS
End podprogramu WektorJednostkowy
```

Podprogram obliczania jednostkowego wektora normalnego za pomocą metody iloczynu wektorowego może mieć postać taką, jak przedstawiono w tabl.13.2.

Tablica 13.2. Podprogram *Normalna*

```
Subroutine Normalna(X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,;XN,YN,ZN)
Real X1, X2, X3, XN, Y1, Y2, Y3, YN, Z1, Z2, Z3, ZN
/* Podprogram oblicza jednostkowy wektor normalny umieszczony w punkcie
(X2, Y2, Z2). Zakłada się, że punkty są uporządkowane w kierunku zgodnym
z ruchem wskazówek zegara, gdy patrzy się na przednią stronę płaszczyzny */
Local Real XA, XB, XX, YA, YB, YY, ZA, ZB, ZZ
        XA = X1 - X2
        YA = Y1 - Y2
        ZA = Z1 - Z2
        XB = X3 - X2
        YB = Y3 - Y2
        ZB = Z3 - Z2
/* Oblicz iloczyn wektorowy */
        XX = YB * ZA - ZB * YA + X2
        YY = XA * ZB - XB * ZA + Y2
        ZZ = XB * YA - XA * YB + Z2
/* Przejdź na długość jednostkową */
        Call WektorJednostkowy(X2,Y2,Z2,XX,YY,ZZ;XN,YN,ZN)
End podprogramu Normalna
```

Do obliczania cosinusa kąta między dwoma wektorami o jednostkowej długości można użyć podprogramu IloczynSkalarny, który może być wywoływany jako funkcja, jeżeli system na to pozwala (tabl. 13.3).

Tablica 13.3. Podprogram *IloczynSkalarny*

```
Subroutine IloczynSkalarny(X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3;Cos)
Real X1, X2, X3, Y1, Y2, Y3, Z1, Z2, Z3, Cos
```

```
/* Podprogram oblicza cosinus kąta między dwoma wektorami jednostkowymi
o punkcie zaczepienia (X2, Y2, Z2) */
Local Real XA, XB, YA, YB, ZA, ZB
        XA = X1 - X2
        YA = Y1 - Y2
        ZA = Z1 - Z2
        XB = X3 - X2
        YB = Y3 - Y2
        ZB = Z3 - Z2
/* Oblicz iloczyn skalarny */
        Cos = XA * XB + YA * YB + ZA * ZB
End podprogramu IloczynSkalarny
```

Stosując te programy można sprawdzić, czy dowolna płaszczyzna opisana za pomocą wierzchołków, uporządkowanych w kierunku zgodnym z ruchem wskazówek zegara może być widoczna z określonego punktu obserwacji. Sprawdzenie można przeprowadzić tak, jak to pokazano w tabl. 13.4.

Tablica 13.4.

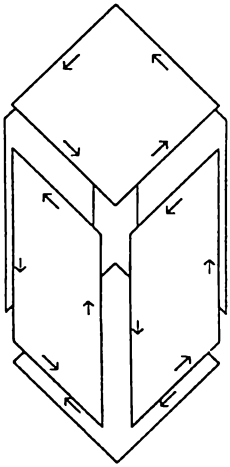
```
/* Fragment programu do sprawdzenia widoczności płaszczyzny opisanej przez
wierzchołki uporządkowane zgodnie z ruchem wskazówek zegara. Program ko-
rzysta z trzech punktów P1(X1,Y1,Z1), P2(X2,Y2,Z2), P3(X3,Y3,Z3), które nie
mogą leżeć na jednej linii prostej. Współrzędne przykładowej testowanej
płaszczyzny są następujące: P1(3,1,4), P2(0,0,0), P3(1,4,3). Punkt obserwacji jest
usytuowany w punkcie (2, 1, -10) */
/* Oblicz normalną */
        Call Normalna(3,1,4,0,0,0,1,4,3;XN,YN,ZN)
/* Oblicz wektor jednostkowy biegnący od P2 do punktu obserwacji */
        Call WektorJednostkowy(0,0,0,2,1,-10;XU,YU,ZU)
/* Oblicz iloczyn skalarny jednostkowego wektora normalnego i wektora
jednostkowego */
        Call IloczynSkalarny(XU,YU,ZU,0,0,0,XN,YN,ZN;Cos)
/* Sprawdź widoczność */
        If Cos > 0 Then
          Widoczna
        Else
          Niewidoczna
        Endif   /* Koniec fragmentu programu */
```

Wyniki są następujące: XN = 0.732467, YN = 0.281718, Zn = –0.61978, XU = 0.19518, YU = 0.09759, ZU = –0.9759, Cos = 0.775299; płaszczyzna jest widoczna. Dla tej samej płaszczyzny, ale przy obserwacji z punktu (–3,0,–3) otrzymamy: XU = –0.7071, YU = 0,

ZU = −0.7071, Cos = −0.079681; płaszczyzna jest niewidoczna. Zazwyczaj punkt obserwacji umieszczamy na ujemnej osi Z; ma to swoje uzasadnienie. Współrzędne punktu obserwacji wynoszą wtedy (0,0,−D), gdzie D jest odległością punktu obserwacji od początku układu współrzędnych.

13.4. Reprezentacja brzegowa obiektów

Reprezentacja brzegowa jest to opis obiektu za pomocą punktów, krawędzi i odpowiednio zorientowanych ścian bocznych, które tworzą obiekt (rys. 13.6). Jest to prosta reprezentacja (być może najprostsza jednoznaczna) wystarczająca dla wielu obiektów, które chcemy narysować. Jednak przygotowanie opisu jest zadaniem pracochłonnym i podatnym na błędy. Do opisu niektórych kształtów nie ma innego sposobu, jak pracowicie, ręcznie wyznaczyć współrzędne, krawędzie i płaszczyzny. Na szczęście inne obiekty, nadają się do zautomatyzowanego opisywania.



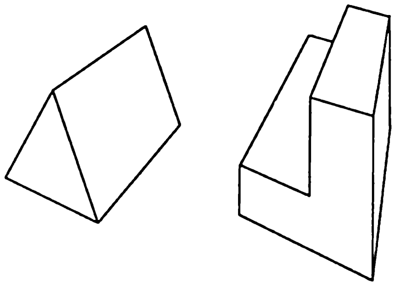
Rys. 13.6. Reprezentacja brzegowa prostopadłościanu

W szczególności są to:

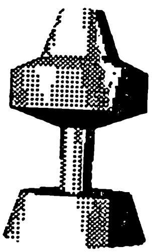
- bryły, które mają jednakowy przekrój na całej długości (rys. 13.7) i które można traktować jako formy wycięte za pomocą wykrojnika o stałym przekroju i przycięte do żądanej długości.
- bryły obrotowe (rys. 13.8), o przekroju, który w wyniku obrotu wokół osi o 360° wyznacza całą formę.

Formy wycięte

Kształt bryły pierwszego rodzaju może być generowany przez podprogram Wycinanie (tabl. 13.5), który tworzy tablice na podstawie szczegółów o przekroju w płaszczyźnie



Rys. 13.7. Wycięte obiekty



Rys. 13.8. Bryła obrotowa (gładkie cieniowanie)

X – Y i głębokości w kierunku Z (ściana przednia jest w płaszczyźnie Z=0). W celu uzyskania formy wyciętej należy utworzyć pierwotny opis obiektu w tej "kanonicznej" pozycji, a następnie dokonać transformacji jego współrzędnych do żądanej pozycji, wymiarów i orientacji za pomocą przesuwania, skalowania i obrotów.

Tablica 13.5. Podprogram *Wycinanie*

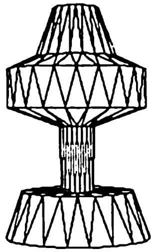
```
Subroutine Wycinanie(Num,Tabl,Głębokość;N,Punkty,Kawałki, Ściany)
Integer Num, N
Real Tabl, Głębokość, Punkty, Kawałki, Ściany
/* Podprogram generuje dane dla wyciętego kształtu opisanego przez
Tabl(Num,2). Otrzymane punkty końcowe są umieszczane w tablicy Punkty(N,3),
a indeksy punktów w tablicach Kawałki(2,Num+1) i Ściany(Num,5) */
Local Integer J
Dim Punkty(2*Num,3), Kawałki(2,Num+1), Ściany(Num,5)
For J = 1 to Num    /* wskazanie tablicy punktów */
        Punkty(J,1) = Tabl(J,1)
        Punkty(J+Num,1) = Tabl(J,1)
        Punkty(J,2) = Tabl(J,2)
        Punkty(J+Num,2) = Tabl(J,2)
        Punkty(J,3) = 0
        Punkty(J+Num,3) = Głębokość
Next J
```

```
/* Najpierw są analizowane ściany końcowe; indeksy punktów są wstawiane do
tablic Kawałki */
For J = 1 to Num
        Kawałki(1,J) = J
        Kawałki(2,J) = 2* Num - (J - 1)
Next J
        Kawałki(1,Num+1) = 1
        Kawałki(2,Num+1) = 2 * Num
/* Teraz przetwarzaj dalsze (Num - 1) ścian */
For J = 1 to Num - 1
        Ściany(J,1) = Num + J
        Ściany(J,2) = Num + J + 1
        Ściany(J,3) = 1 + J
        Ściany(J,4) = J
        Ściany(J,5) = Num + J
Next J
/* Wreszcie analizuj ostatnią ścianę */
        Ściany(Num,1) = 2 * Num
        Ściany(Num,2) = Num + 1
        Ściany(Num,3) = 1
        Ściany(Num,4) = Num
        Ściany(Num,5) = 2 *Num
End podprogramu Wycinanie
```

Dla obiektów o złożonych przekrojach są potrzebne dwie tablice. Dla bryły o przekroju będącym 20-kątem jest potrzebna tablica 40×4 dla współrzędnych i lista odcinków o 164 elementach, nie licząc tablicy 20×2 do pierwotnego opisu. Aproksymacja walca za pomocą 72 boków, wymaga tablicy 144×4 dla współrzędnych, listy odcinków o 580 pozycjach i tablicy 72×2 z opisem pierwotnym. Jeżeli tablice o takich wymiarach stwarzają problemy, to można przyjąć, że obiekt zajmuje cały segment. Stosując metody przedstawione w rozdz. 11, można zapisywać informacje na dysku zaraz po ich utworzeniu. Informacje potrzebne na bieżąco mogą być potem odczytane z dysku do chwilowej, małej tablicy, niezależnie dla każdej płaszczyzny. Jeżeli dysk nie jest dostępny, to nie można w wygodny sposób zapamiętać współrzędnych i każda ściana jest rysowana od razu po utworzeniu.

Formy obrotowe

Łatwo można opisywać złożone formy symetryczne (rys. 13.9), które uzyskuje się przez obrót figury o 360°, krokami po $360^\circ/n$ (n jest liczbą ścian bryły). Po systematycznym zdefiniowaniu wszystkich punktów wierzchołków całości, należy sporządzić listę krawędzi jednej ściany i powtórzyć to dla wszystkich pozostałych ścian. Należy pamiętać, żeby krawędzie każdego wielokąta były uporządkowane w kierunku zgodnym z ruchem wskazówek zegara po to, żeby inne podprogramy mogły obliczyć zewnętrzne nor-



Rys. 13.9. Bryła obrotowa (złożona ze ścianek)

malne. Gdy jest spełniony ten warunek jest możliwe rysowanie w perspektywie tylko tych ścian, które są aktualnie widoczne. W Dodatku 2 podano program dla mikrokomputera Apple Macintosh, pozwalający na wyświetlanie form obrotowych w sposób wyżej opisany.

Podprogram Wycinanie nadaje się zarówno do opisu prostych kształtów, takich jak prostopadłościany, jak i bardziej złożonych brył. Ponieważ prostopadłościan często występuje w obrazach grafiki komputerowej, warto opracować dla tego obiektu specjalny, szybki program określania współrzędnych i ścian. Należy zdefiniować prostopadłościan w położeniu "kanonicznym", a potem przekształcić jego współrzędne tak, żeby uzyskać właściwe położenie, orientację i wielkość. W celu uzyskania dowolnego prostopadłościanu, wystarczy opisać sześcian jednostkowy o środku w początku układu współrzędnych.

ZADANIA

13.1. W rozdziale zasugerowaliśmy sposoby reprezentacji współrzędnych wierzchołków i ścian dla trójwymiarowych obiektów. Podaj zalety i wady każdej metody i spróbuj zaproponować jeszcze inne metody. Należy pamiętać, że każda reprezentacja musi zapewniać kompromis między: wymaganiami dotyczącymi łatwości korzystania, nakładów obliczeniowych, wykorzystaniem pamięci (zajętość) i wiarygodnością opisu. W szczególności reprezentacja musi być jednoznaczna, tzn. w każdym przypadku reprezentacja musi opisywać tylko jedną bryłę.

13.2. Sprawdź twierdzenie Eulera o wielościanach, na bryłach z najbliższego otoczenia. Czy istnieją bryły, dla których formuła zawodzi? W podanej postaci, twierdzenie nie obejmuje przypadków brył z dziurami. Zastanów się nad sposobem zmodyfikowania formuły tak, żeby obejmowała ona również i takie przypadki (jest to dość trudne, ale wykonalne). Co można powiedzieć o kulach i walcach?

13.3. Napisz podprogram wykorzystujący podaną metodę wyznaczania współczynników równania płaszczyzny określonej przez trzy punkty P1=(X1,Y1,Z1), P2=(X2,Y2,Z2) i P3=(X3,Y3,Z3). Pamiętaj o sprawdzaniu, czy wszystkie te punkty nie leżą na jednej prostej. Korzystając z podprogramu, sprawdź następujące wyniki:

P1 = (2,4,6), P2 = (0,0,0), P3 = (5,2,7)

A = –16, B = –16, C = 16, D = 0

P1 = (5,5,5), P2 = (–2,7,0), P3 = (0,0,1)

A = –33, B = –3, C = 45, D = –45
P1 = (0,0,0), P2 = (10,4,7), P3 = (1,2,3)
A = –2, B = –23, C = 16, D = 0
P1 = (1,1,1), P2 = (4,1,0), P3 = (5,1,2)
A = 0, B = –7, C = 0, D = 7
P1 = (1,1,1), P2 = (2,1,3), P3 = (3,1,5)
Punkty kolinearne nie wyznaczają jednej wyróżnionej płaszczyzny.

13.4. Zauważ, że wartości A, B, C i D nie są unikalne. Dowolne wartości, między którymi zachodzą te same proporcje będą równie dobre. Stąd współczynniki płaszczyzny do pierwszego pytania mogą być wyrażone jako (–1,–1,1,0) (po podzieleniu przez 16) albo nawet (1,1,–1,0) (po podzieleniu przez –16). Tak więc równanie płaszczyzny ma postać X+Y–Z=0. Co te wyniki mówią Tobie o płaszczyźnie, która przechodzi przez początek układu współrzędnych? (Wskazówka: wartość D jest istotna). Czy to, co było powiedziane, wyjaśnia dlaczego współczynnik D nie jest używany do obliczania normalnej?

13.5. Skorzystaj z wyników zadania 13.3 do sprawdzenia, że normalne dla każdej płaszczyzny w punkcie P2 są następujące:
P2 = (0,0,0) PN = (–16,–16,16)
P2 = (–2,7,0) PN = (–35,4,45)
P2 = (10,4,7) PN = (8,–19,23)
P2 = (4,1,0) PN = (4,–6,0)

13.6. Pokaż, że dla danych płaszczyzn jednostkowe wektory normalne w tych punktach są następujące:
od (0,0,0) do (–1,–1,1), UN = (–0.57735,–0.57735, 0.57735);
od (–2,7,0) do (–34,4,45), Un = (–2.59051, 6.94632, 0.805242);
od (10,4,7) do (8,–19,23), Un = (9.9288, 3.18118, 7.56962);
od (4,1,0) do (4,–6,0), UN = (4,0,0).

Ze względu na sposób obliczania jednostkowego wektora normalnego, nie powinno być zaskoczeniem, że w większości przypadków jego składowe będą raczej liczbami rzeczywistymi niż całkowitymi. Co możesz powiedzieć o płaszczyźnie, dla której wektor normalny wynosi (4,0,0)?

13.7. Wykorzystaj opracowany podprogram do sprawdzenia, że kolejność podania trzech punktów jest ważna i po jej odwróceniu, normalna będzie skierowana w przeciwnym kierunku. Przed ostatnimi obliczeniami, tymi z jednostkowym wektorem normalnym (4,0,0), zgadnij jaki powinien być wynik. (Wskazówka: pomóc powinien szkic).

13.8. W Dodatku 1 jest podany alternatywny sposób obliczania normalnej do powierzchni za pomocą iloczynu wektorowego, wykorzystany w tabl. 13.2. Napisz podprogram Normalna i wykorzystaj go do sprawdzenia wyników zadania 13.5. Pamiętaj o tym, że punkty powinny być przekazywane do podprogramu w ustalonym porządku i wektor od (X0,Y0,Z0) do (X9,Y9,Z9) ma postać (X9 – X0, Y9 – – Y0, Z9 – Z0). Potwierdź, że do otrzymania takich samych wyników jak w zadaniu 13.5, punkty powinny być podane w przeciwnej kolejności.

13.9. Korzystając z opracowanej wersji podprogramu IloczynSkalarny, sprawdź, że cosinus kąta między wektorami jednostkowymi (-0.57735, -0.57735, 0.57735) i (0,0,-1) wynosi -0.57735. Ponieważ pierwszy wektor jednostkowy jest prostopadły do płaszczyzny $X + Y - Z = 0$, określ czy płaszczyzna jest widoczna, czy nie, z punktu (0,0,-1)?

13.10. Napisz podprogram Wycinanie i użyj go do opisu punktów i płaszczyzn o różnych kształtach. Zmodyfikuj program dla komputera Macintosh podany w Dodatku 2 tak, żeby mógł być użyty w dostępnym Tobie komputerze i użyj tego programu do tworzenia brył obrotowych.

13.11. Napisz i przetestuj podprogram rysowania prostopadłościanu. Porównaj jego działanie z wynikami otrzymanymi przez wycinanie prostokątnego przekroju.

14. Transformacje trójwymiarowe

W stosunku do obiektów trójwymiarowych można stosować podobnie jak w przypadku obiektów dwuwymiarowych transformacje: przesunięcia, obrotu, skalowania i odkształcania. Ponadto są znane inne transformacje takie, jak *przekształcenie perspektywiczne* pozwalające przejść z reprezentacji trójwymiarowej na płaską, co umożliwia przedstawienie na płaszczyźnie obiektów przestrzennych, obserwowanych z określonego punktu.

14.1. Przesunięcie

Do przesunięcia punktu (Xp, Yp, Zp, 1) o wartości Xmv, Ymv i Zmv, korzystamy z macierzy T:

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ Xmv & Ymv & Zmv & 1 \end{bmatrix}$$

i otrzymujemy współrzędne nowego punktu:

(Xn, Yn, Zn, 1) = (Xp, Yp, Zp, 1) * T = (Xp + Xmv, Yp + Ymv, Zp + Zmv, 1)

14.2. Obrót

W przypadku dwuwymiarowym była potrzebna tylko transformacja obrotu w płaszczyźnie X-Y wokół domniemanej osi Z. W przypadku trzech wymiarów mamy trzy osie, wokół których są możliwe obroty. (Obroty wokół dowolnej prostej można zrealizować przez połączenie przesunięcia i obrotów wokół osi).

- Obrót o kąt A wokół ujemnej półosi Z - równoważny obrotowi w płaszczyźnie (był już opisany) - opisuje macierz **Oz**

$$Oz = \begin{bmatrix} C & S & 0 & 0 \\ -S & C & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

gdzie: C = cos(A) i S = sin(A).
Współrzędne obróconego punktu wynoszą:

(Xn,Yn,Zn, 1) = (Xp,Yp,Zp, 1) * Oz =
= (Xp * C – Yp * S, Xp * S + Yp * C, Zp, 1)

- Obrót wokół dodatniej półosi X opisuje macierz **Ox**

$$Ox = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C & -S & 0 \\ 0 & S & C & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Współrzędne obróconego punktu wynoszą:

(Xn, Yn, Zn, 1) = (Xp, Yp, Zp, 1) * Ox =
= (Xp, Yp * C + Zp * S, Zp * C – Yp * S, 1)

- Obrót wokół dodatniej półosi Y określa macierz **Oy**

$$Oy = \begin{bmatrix} C & 0 & S & 0 \\ 0 & 1 & 0 & 0 \\ -S & 0 & C & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Współrzędne obróconego punktu wynoszą:

(Xn, Yn, Zn, 1) = (Xp, Yp, Zp, 1) * Oy =
= (Xp * C – Zp * S,Yp, Xp * S + Zp * C, 1)

Jeżeli będziemy dokonywali obrotów dla dodatnich kątów, to z punktu leżącego na ujemnej osi Z zaobserwujemy, że w przypadku:

Ox: obiekt pochyla się w naszą stronę
Oy: prawa strona obiektu oddala się od nas
Oz: obiekt obraca się w kierunku przeciwnym do ruchu wskazówek zegara.

Do utworzenia odpowiedniej macierzy korzystamy z podprogramu MacierzObrotu podanego w tabl. 14.1.

Tablica 14.1. Podprogram *MacierzObrotu*

```
Subroutine MacierzObrotu(Oś,A;Mac)
String Oś
Real A, Mac
```

```
/* Podprogram tworzy macierz obrotu Mac dla obrotu o kąt A (podany w stop-
niach). Oś, wokół której ma być dokonany obrót jest określana przez wstawienie
w miejsce 'Oś': 'x' albo 'y' albo 'z' */
Local Integer H, J
Local Real C, S
Dim Mac(4,4)
         A = A * PI/180
         C = Cos(A)
         S = Sin(A)
         For H = 1 to 4    /* inicjacja macierzy */
            For J = 1 to 4
               If H = J Then
                  Mac(H,J) = 1
               Else
                  Mac(H,J) = 0
               Endif
            Next J
         Next H
         Case Oś of:
            'x':Mac(2,2) = C
               Mac(2,3) = - S
               Mac(3,2) = S
               Mac(3,3) = C
            'y':Mac(1,1) = C
               Mac(1,3) = S
               Mac(3,1) = - S
               Mac(3,3) = C
            'z':Mac(1,1) = C
               Mac(1,2) = S
               Mac(2,1) = - S
               Mac(2,2) = C
End podprogramu MacierzObrotu
```

14.3. Skalowanie

Skalowanie pozycji punktu (Xp,Yp,Zp,1) o wielkość Xsk, Ysk, Zsk opisuje macierz

$$S = \begin{bmatrix} Xsk & 0 & 0 & 0 \\ 0 & Ysk & 0 & 0 \\ 0 & 0 & Zsk & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

W wyniku skalowania otrzymujemy współrzędne nowego punktu:

(Xn,Yn,Zn, 1) = (Xp,Yp,Zp, 1) * S = (Xp * Xsk, Yp * Ysk, Zp * Zsk, 1)

14.4. Transformacje złożone

Podobnie jak w przypadku dwuwymiarowym można kolejno stosować transformacje przesunięcia, obrotu i skalowania; można również połączyć te macierze w jedną i z niej korzystać. (Alternatywnie można korzystać z bezpośredniej reguły transformacji jeżeli nie chcemy w ogóle używać macierzy). Jeżeli zdecydujemy się na łączenie macierzy, to będzie potrzebny podprogram mnożenia dwóch macierzy, w celu uzyskania trzeciej. Służy do tego podprogram MnożMac. Ważna jest kolejność mnożenia macierzy: iloczyn Mac1xMac2 rzadko daje ten sam wynik co iloczyn Mac2xMac1. Na przykład jeżeli najpierw obrócimy punkt (3,3,3,1) o 30° wokół osi X i potem o 30° wokół osi Y, to znajdziemy się w punkcie (2.049,4.098,2.4509,1). To jest równoważne użyciu M3 w stosunku do punktu (3,3,3,1) po wywołaniu podprogramu MnożMac (Ox,Oy;M3). Jeżeli najpierw dokonamy obrotu o 30° wokół osi Y, a potem o 30° wokół osi X, to otrzymamy (1.098,4.647,2.0489,1). Jest to równoważne użyciu M3 po wywołaniu podprogramu MnożMac (Oy,Ox;M3). A więc pierwszy parametr w MnożMac wskazuje pierwszą transformację, która ma być wykonana, a drugi parametr drugą transformację (tabl. 14.2).

Tablica 14.2. Podprogram *MnożMac*

```
Subroutine MnożMac(M1,M2;M3)
Real M1, M2, M3
/* Podprogram mnoży dwie macierze 4 x 4 w celu uzyskania trzeciej */
Local Integer H, J, K
Local Real Pomoc
Dim M3(4,4)
      For H = 1 to 4
        For J = 1 to 4
          Pomoc = 0
          For K = 1 to 4
            Pomoc = Pomoc + M1(H,K) * M2(K,J)
          Next K
          M3(H,J) = Pomoc
        Next J
      Next H
End podprogramu MnożMac
```

Podprogram StosMac (tabl. 14.3) pozwala użyć macierz transformacji 4×4 do zbioru współrzędnych wierzchołków.

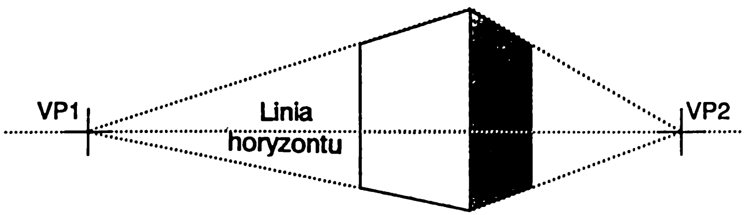
Tablica 14.3. Podprogram *StosMac*

```
Subroutine StosMac(N,CA,Mat;N,OA)
Integer N
Real CA, Mat, OA
```

```
/* Podprogram mnoży każdy z N jednorodnych wektorów w CA przez macierz
transformacji Mac 4x4. Nowe wartości są wstawiane do tablicy OA */
Local Integer H, J, K
Local Real Pomoc
Dim OA(N,4)
      For H = 1 to N
        For J = 1 to 4
          Pomoc = 0
          For K = 1 to 4
            Pomoc = Pomoc + CA(H,K) * Mat(K,J)
          Next K
          OA(H,J) = Pomoc
        Next J
      Next H
End podprogramu StosMac
```

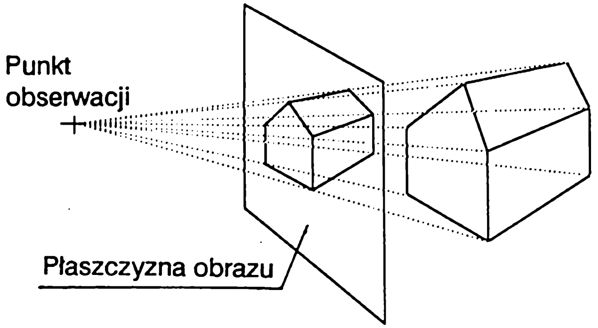
14.5. Perspektywa

W celu realistycznego przedstawienia obiektów należy stosować metody rysowania perspektywicznego. W konwencjonalnych ręcznych rysunkach perspektywa polega na prawidłowym ustawieniu *punktów zbiegu* - punktów, przez które muszą przechodzić wszystkie linie równoległe do danych płaszczyzn (rys. 14.1). W grafice komputerowej, jak można się spodziewać, do uzyskania perspektywy korzysta się z transformacji matematycznych, a nie z punktów zbiegu. Podobnie jak w innych, dotychczs napotkanych transformacjach, stosujemy *transformacje perspektywiczne* do każdego wierzchołka obiektu, który ma być wyświetlany. Transformacja ta zamienia każdy punkt w przestrzeni trójwymiarowej na odpowiednią pozycję na płaszczyźnie.



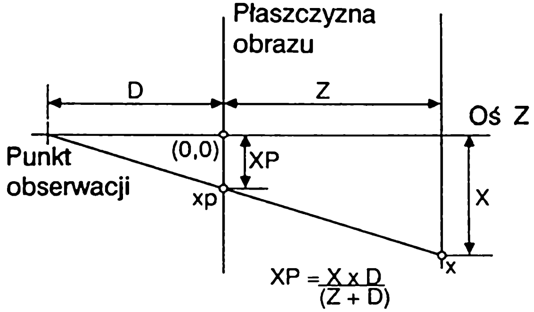
Rys. 14.1. Konwencjonalna perspektywa z dwoma punktami zbiegu

Chcąc zrozumieć, jak wyprowadza się transformację perspektywiczną, wyobraźmy sobie, że patrzymy na obiekt przez szklaną płytkę. Na płytce tej można zaznaczyć punkty odpowiadające poszczególnym wierzchołkom sceny (rys. 14.2). Załóżymy,



Rys. 14.2. Linie biegnące od sceny do oka, przecinają płaszczyznę obrazu

że punkt obserwacji znajduje się na ujemnej półosi Z, w odległości D jednostek współrzędnych świata od początku układu współrzędnych i płytka szklana (nazywana *płaszczyzną obrazu*) jest umieszczona w płaszczyźnie XY w początku układu współrzędnych



Rys. 14.3. Obliczanie współrzędnej punktu xp z podobieństwa trójkątów

(rys. 14.3). Przy tym założeniu, każdy punkt o współrzędnej X w scenie ma w płaszczyźnie obrazu współrzędną XP.
Z podobieństwa trójkątów wynika, że

$$XP = X * D/(Z+D) = X * 1/((Z/D)+1)$$

oraz

$$YP = Y * D/(Z+D) = Y * 1/((Z/D)+1)$$

Ponieważ płaszczyzna obrazu jest umiejscowiona w Z = 0, wszystkie współrzędne Z punktów sceny, w obrazie przyjmują wartość zero. Stąd wartości X i Y w płaszczyźnie obrazu zależą od wartości X i Y w przestrzeni i od współczynnika zależnego od odległości oka i od wartości Z w przestrzeni. Tak więc
(Xn, Yn, Zn, 1) = (Xp * F,Yp * F,0,1); gdzie F = 1/((Zp/D)+1)
W zapisie macierzowym współrzędne punktów obrazu wynoszą:

$$(Xn, Yn, Zn, 1) = (Xp, Yp, Zp, 1) * P;$$

gdzie

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/D \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Stosując transformację opisaną macierzą **P** do punktu (Xp,Yp,Zp, 1), otrzymamy punkt (Xp,Yp,0,Zp/D + 1). Pamiętając o tym co było powiedziane o współrzędnych jednorodnych, można powyższy zapis przedstawić w równoważnej postaci:

(Xp · F, Yp · F, 0, 1);

gdzie F zostało wcześniej zdefiniowane.

Do utworzenia perspektywicznego widoku obrazu, wykonujemy następujące czynności:

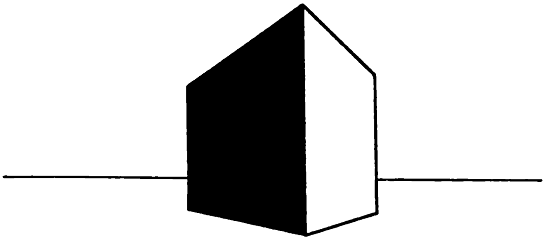
1. Przesuwamy obiekt do pozycji odpowiedniej dla określonego punktu obserwacji.
2. Obracamy obiekt tak, żeby widoczne były właściwe ściany.
3. Wykonujemy transformację perspektywiczną na współrzędnych wierzchołków, w celu odwzorowania ich na płaszczyznę obrazu.
4. O ile zachodzi taka potrzeba skalujemy obraz tak, żeby rysunek odpowiadał wielkości powierzchni do rysowania.
5. Wyświetlamy wynik, łącząc odcinkami (ewentualnie obciętymi) punkty otrzymane po transformacji.

Tak, jak poprzednio transformacje można wykonywać niezależnie lub łącznie. Macierze przesunięcia, obrotu i skalowania są takie same dla wszystkich współrzędnych obiektu, natomiast macierz perspektywy zależy od wartości współrzędnej Z w każdym punkcie. Zatem jest wygodnie łączyć macierze transformacji i obrotu, natomiast macierze skalowania i perspektywy stosować osobno. Ponieważ transformacja perspektywiczna dotyczy tylko współrzędnych X i Y, może się okazać, że lepiej jest w tym przypadku stosować bezpośrednią formułę konwersji niż zapis macierzowy.

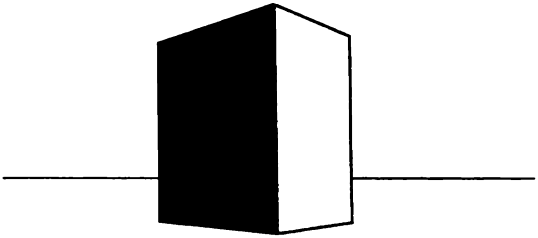
14.6. Zmiana widoku

Najlepszy widok sceny możemy uzyskać dobierając wartości kilku różnych parametrów:

- Mając okno o ustalonej szerokości, można przybliżać lub oddalać punkt obserwacji od płaszczyzny obrazu. Innymi słowy, można zmieniać wartość D. Dla punktu obserwacji znajdującego się stosunkowo blisko płaszczyzny obrazu uzyskuje się taki obraz jak na fotografii otrzymanej za pomocą szerokokątnego obiektywu; odległe obiekty wydają się małe; bliskie wydają się duże; kąty narożników bliskich punktowi obserwacji są bardzo ostre (rys. 14.4). Jest to metoda otrzymywania "dramatycznych" widoków. Odwrotnie, gdy punkt obserwacji jest stosunkowo daleko od płaszczyzny obrazu, uzyskuje się efekt taki, jak przy fotografowaniu za pomocą teleobiektywu (rys. 14.5). Do uzyskania mniej lub bardziej "normalnego" wyglądu, kąt widzenia powinien wynosić około 60°. Oznacza to, że punkt obserwacji powinien być z grubsza oddalony od płaszczyzny obrazu, o odległość równą większemu z wymiarów: szerokości lub wysokości okna.



Rys. 14.4. Perspektywa szerokokątna

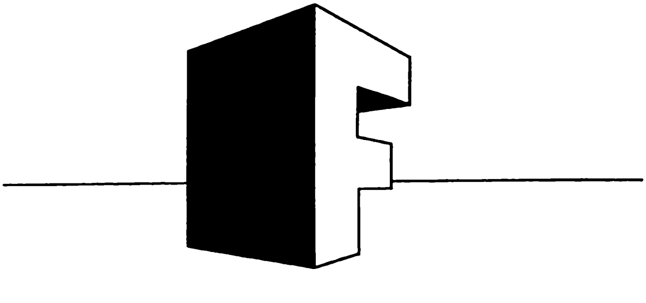


Rys. 14.5. Perspektywa "teleobiektywowa"

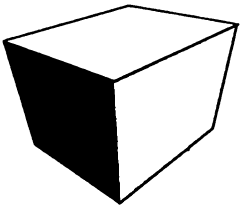
Podobne efekty można otrzymać przy stałej odległości punktu obserwacji i zmianie wymiarów okna. Stosunkowo wąskie okno daje efekt szerokiego kąta.

● Można obracać obiekty wokół dowolnej osi. Zwykle obiekt jest obracany wokół osi Y w celu uwidocznienia wybranych ścian (rys. 14.6). Jeżeli dodatkowo należy uwidocznić górę lub dół obiektu, to należy obrócić go o dodatni kąt wokół osi X tak, że - by zobaczyć górę, albo o ujemny kąt tak, żeby zobaczyć dół. Zauważymy wtedy (rys. 14.7), że linie pionowe stają się ukośne. One również zbiegają się do punktu zbiegu i to pomaga w uzyskaniu żądanego efektu dramatyzmu. Obrót wokół osi Z pochyla obiekt na bok i z wyjątkiem konieczności uzyskania specjalnych widoków jest rzadko stosowany.

● Obrazy mogą być przesuwane względem płaszczyzny obrazu. To oznacza zwiększenie lub zmniejszenie ich współrzędnych przez przesunięcie. Zwiększanie współrzędnych X przesuwa obiekt w prawo, a zmniejszanie w lewo. Zwiększanie współrzędnych



Rys. 14.6. Litera obrócona wokół osi Y w celu uwidocznienia wybranych ścian



Rys. 14.7. Pochylenie wokół osi X wprowadza trzeci punkt zbiegu

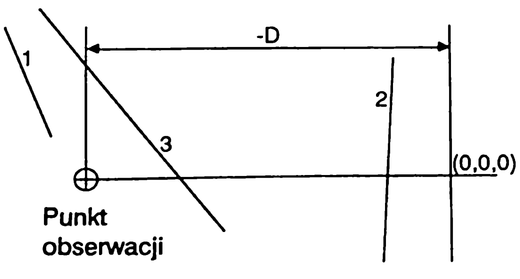
Y przesuwa obiekt do góry co uwidocznia dolne powierzchnie. Zmniejszanie współrzędnych Y przesuwa obiekt w dół i uwidacznia górę obiektu. Jest to alternatywne działanie w stosunku do obracania wokół osi X; nie osiąga się jednak zbieżności linii pionowych. Zwiększanie współrzędnych Z oddala obiekt; zmniejszanie współrządnych Z powoduje przybliżenie obiektu.

● Przy zmniejszaniu współrzędnej Z należy zachować ostrożność, żeby nie przesunąć żadnej części obiektu poza punkt obserwacji. W takim przypadku mogłyby pojawić się dziwne efekty. Jeżeli symulujemy efekt poruszania się po scenie, to pewne części obiektu mogą przypadkowo znaleźć się z tyłu za obserwatorem. Ghcąc uniknąć związanych z tym konsekwencji, należy wprowadzić inny rodzaj obcinania: Z - obcinanie, które powoduje usunięcie linii lub ich części, znajdujących się za punktem obserwacji. Działania te należy wykonać po obrotach i przesunięciach, ale przed transformacją perspektywiczną.

14.7. Z - obcinanie

Jeżeli okno znajduje się w odległości –D na osi Z, to można obcinać za pomocą płaszczyzny Z = –D. W tym celu należy sprawdzić wartości współrzędnej Z każdego odcinka, który będzie rysowany. Tak jak pokazano na rys. 14.8, w trakcie obcinania mogą powstać trzy sytuacje:

● współrzędne Z obu punktów końcowych są mniejsze lub równe –D i cały odcinek jest poza punktem obserwacji; zatem można go pominąć,



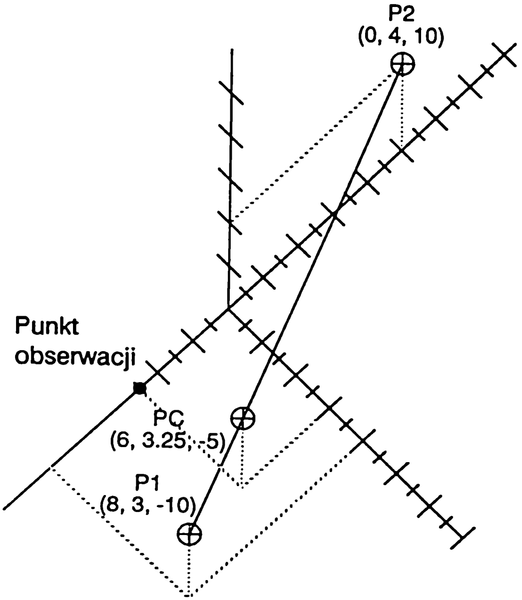
Rys. 14.8. Z - obcinanie odcinków;
1 - odcinek ten można pominąć ponieważ całkowicie leży poza punktem obserwacji,
2 - odcinek jest widoczny i powinien podlegać Z - obcięciu,
3 - część tego odcinka leży poza punktem obserwacji i musi być obcięta

- współrzędne Z obu punktów obserwacji są większe niż –D i cały odcinek jest przed punktem obserwacji; zatem można go nie obcinać,
- współrzędna Z jednego punktu końcowego jest większa niż –D, a druga mniejsza lub równa –D; wówczas należy dokonać obcięcia. W tym przypadku wystarczy podstawić wartość Z = –D do równania odcinka.

Dla odcinka łączącego punkty P1 (8,3,-10) i P2 (0,4,10) i okna w odległości 5 jednostek wzdłuż ujemnej półosi Z (rys. 14.9), punkt P1 znajduje się za punktem obserwacji, i punkt P2 znajduje się z przodu. Równania parametryczne tego odcinka są następujące:

$$X = 8 + t * (0\text{-}8) = 8 - 8 * t$$
$$Y = 3 + t * (4\text{-}3) = 3 + t$$
$$Z = -10 + t * (10-(-10)) = -10 + 20 * t$$

Wówczas, gdy Z = –5, t = 0.25 i X = 6, Y = 3.25. Obcięty odcinek łączy punkty PC (6,3.25,-5) i P2 (0,4,10); te punkty należy przekazać do podprogramu perspektywy.



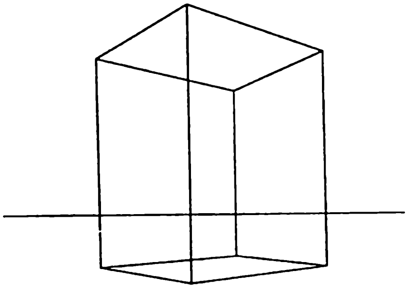
Rys. 14.9. Obcięcie przez płaszczyznę zawierającą punkt obserwacji

14.8. Linie i powierzchnie niewidoczne

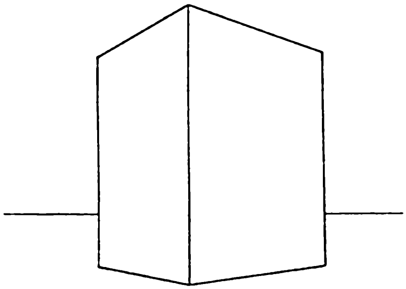
Jeżeli opisujemy obiekty jako zbiory punktów i odcinków, to można rysować je w postaci szkieletowej (rys. 14.10). Takie rysunki nie są zbyt realistyczne i dla złożonych scen mogą być bardzo zagmatwane; są one jednak dość często stosowane w animacji. Wynika to z faktu, że nawet mały ruch wprowadza paralaksę sugerującą głębokość

obiektu. Do uzyskania bardziej przekonującego obrazu ciągłości struktury bryły, trzeba użyć minimalnej formy reprezentacji brzegowej opisując punkty, odcinki i płaszczyzny. Korzystając z metody podanej w tabl. 13.4, można sprawdzić, czy płaszczyzny tworzące obiekt są skierowane do obserwatora i narysować je, jeżeli są widoczne.

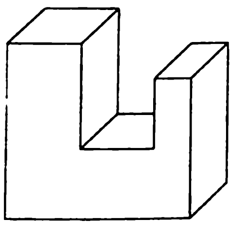
W przypadku pojedynczych wypukłych obiektów ściany mogą być widoczne w całości, bądź wcale. Dlatego w takich przypadkach wystarczy najpierw wykonać test widoczności, a następnie narysować widoczne ściany w celu uzyskania wrażenia pełności bryły (rys. 14.11). Jeżeli w scenie znajduje się więcej niż jeden obiekt lub jeżeli pojedynczy obiekt nie jest wypukły (tak jak ten z rys. 14.12), to powierzchnie, które są skierowane do obserwatora mogą być w całości lub częściowo zasłonięte przez inne ciała lub części samego obiektu. Podczas realizacji takich przypadków należy korzystać z bardziej wyszukanych metod opisu obiektu.



Rys. 14.10. Rysunek szkieletowy



Rys. 14.11. Ściany zasłaniają widok innych części obiektu



Rys. 14.12. Obiekt niewypukły

Stosowane metody różnią się między sobą; można rysować tylko linie, uwidaczniając krawędzie płaszczyzn, lub obszary wypełnione, uwidaczniając powierzchnie. W pierwszym przypadku należy rozważyć problem linii niewidocznych, w drugim - problem powierzchni niewidocznych. Zaskakujący jest fakt, że łatwiej jest rozwiązywać problem niewidocznych powierzchni niż niewidocznych linii.

Jedna, szczególnie prosta metoda usuwania niewidocznych powierzchni jest znana jako *algorytm malarski*, nazwana tak, gdyż przypomina sposób malowania obrazów (chociaż w zasadzie żaden artysta nie używa tak osobliwej metody). Malując obraz farbami olejnymi, najpierw zamalowuje się tło, tak jakby nie było w scenie żadnych obiektów. Następnie są malowane obiekty najbardziej od nas oddalone, potem trochę bliższe obiekty itd., na końcu są tworzone obiekty leżące najbliżej oka. W czasie tego procesu stopniowo są dodawane do sceny kompletne obiekty w kolejności uzależnionej od ich odległości od nas i za każdym razem zakłada się, że nie ma żadnych bliższych obiektów do pokazania. W miarę tworzenia obrazu, bliższe obiekty będą automatycznie zasłaniały części dalszych obiektów i po zakończeniu procesu uzyska się właściwe uporządkowanie obiektów.

Do zrealizowania tej metody za pomocą komputera należy:

- Usunąć z rozważań wszystkie płaszczyzny skierowane przeciwnie do punktu obserwacji, ponieważ nie mogą być widoczne. Umożliwia to metoda podana w tabl. 13.4.
- Posortować pozostałe ściany ze względu na odległości od punktu obserwacji zaczynając od ściany najbardziej odległej.
- Uzupełniać kolorem ściany w kolejności zgodnej z posortowaną listą.

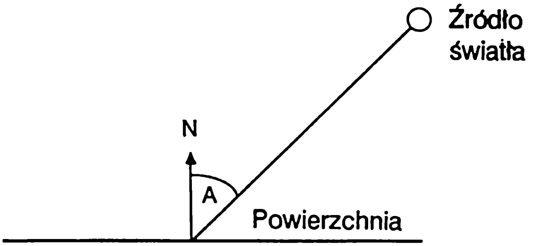
14.9. Cieniowanie

Kolejnym elementem zwiększenia realizmu kolorowych rysunków jest wprowadzenie cieniowania ścian obiektów, w zależności od ilości padającego na nie światła. Oczywiście te boki obiektu, które są skierowane bezpośrednio w stronę źródła światła wydają się być jaśniejsze niż te, na które światło pada ukośnie. Niestety stopień jasności nie jest bezpośrednio związany z kątem padania światła. Prawo znane w fizyce jako *prawo Lamberta* mówi, że ilość światła odbitego od punktu na powierzchni matowej jest proporcjonalna do cosinusa kąta między źródłem światła a normalną do powierzchni w tym punkcie (rys. 14.13).

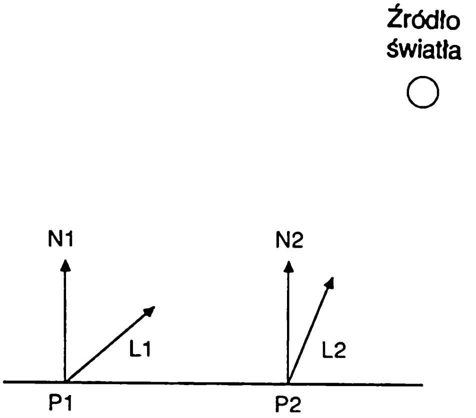
Cosinus kąta między dwoma wektorami normalnymi jest określony przez iloczyn skalarny (rozdz. 13); jeśli więc jest dane położenie źródła światła i normalna do powierzchni, to można policzyć jasność koloru, jaki należy przypisać powierzchni. Można tu korzystać z formuły

Jasność = K * (N · L);

gdzie: K jest stałą, za pomocą której można wpływać na wartości jasności, (N · L) jest iloczynem skalarnym jednostkowego wektora normalnego do powierzchni (N) i wektora jednostkowego skierowanego do źródła światła (L). Jeżeli źródło światła znajduje się dość blisko powierzchni, to można zauważyć (rys. 14.14), że L będzie różne



Rys. 14.13. Oświetlenie zależy od cosinusa kąta między normalną do powierzchni a promieniem ze źródła światła



Rys. 14.14. Gdy źródło światła jest blisko, wektory skierowane do niego są różne w różnych punktach powierzchni

w różnych punktach. (Jest to jedna z przyczyn, dla których lampa stołowa nie oświetla całej powierzchni roboczej równomiernie). Jeżeli jednak źródło światła, z którego korzystamy znajduje się daleko od powierzchni - tak jak słońce - to można założyć, że L jest stałe dla całej sceny. Przy takim założeniu, dla dowolnej płaszczyzny wystarczy obliczyć tylko jej wektor normalny, a następnie jego iloczyn skalarny ze stałym wektorem jednostkowym L - co pozwala na wyznaczenie oświetlenia odtwarzanej sceny.

Prawo Lamberta jest słuszne dla światła rozproszonego (takiego jakie jest w pochmurny dzień) i dla powierzchni matowych; nie uwzględnia ono światła otoczenia tzn. takiego, jakie dociera od otaczającej nas sceny. Opisuje ono model warunków oświetlenia w przestrzeni kosmicznej, gdzie nie ma oświetlenia od nieba, a tylko od odległych źródeł światła. Skutkiem odtwarzania oświetlenia powierzchni obiektu w grafice komputerowej przy wykorzystaniu prawa Lamberta, jest przesadne zróżnicowanie oświetlenia powierzchni skierowanej bezpośrednio do światła i powierzchni usytuowanych ukośnie do światła. Wadę tę można usunąć zwiększając jasność poszczególnych punktów, o stałą wartość O reprezentującą wpływ otoczenia. Otrzymujemy w ten sposób zależność

$$\text{Jasność} = O + K * (N \cdot L)$$

Jeżeli chcemy również uzależnić jasność od źródła (Iźródła), to można użyć formuły:

Jasność = Iźródła * O + Iźródła * K * (N · L)

Jeżeli chcemy, żeby obiekt był oświetlony przez światło w proporcjach: 20% - wpływ otoczenia i 80% - oświetlenie bezpośrednie, to można użyć współczynnika otoczenia O = 0.2 i K = 0.8.

Do odtwarzania sceny z rozświetleniem i odbiciami obiektów, należy wziąć pod uwagę inne czynniki, które nie zostały omówione w tej książce (zob. zalecaną literaturę).

ZADANIA

14.1. Napisz i przetestuj programy MacierzObrotu, MnożMac, StosMac. Wykorzystaj je do sprawdzenia obrotów punktu (3,3,3,1) podanego w tekście. Wykonaj próby z kilkoma różnymi punktami, żeby się przekonać, że kolejność wykonywania obrotów jest ważna.

14.2. Sprawdź, czy szybsze jest korzystanie z bezpośredniego mnożenia, czy z mnożenia macierzowego. Dokonaj prób bez i z przesunięciami. Jeżeli jest wykorzystywany język APL albo wersja Basicu z wbudowanymi operacjami macierzowymi (występują w oryginalnym Basicu z Dartmouth), to skorzystaj z tych możliwości języka.

14.3. Napisz i przetestuj podprogram prezentacji obrazu w perspektywie i wykorzystaj go do opisu obiektu uzyskanego za pomocą podprogramów Prostopadłościan i Wycinanie. Przeprowadź eksperymenty z różnymi odległościami obserwacji i wymiarami okna. Kolejno wykonaj obroty wokół każdej z osi. Pokaż, że przy obrocie wokół osi X, linie pionowe przestają wyglądać jak pionowe.

14.4. Napisz i przetestuj program Z-obcinanie wykorzystując opisaną w tekście metodę parametryczną. Zaproponuj wersję dla wielokątów. Zauważ, że program musi określić, który koniec obciętej linii jest widoczny. W obciętym wielokącie, czasami pierwszy punkt odcinka jest widoczny, a drugi jest niewidoczny; czasami zaś odwrotnie. Pamiętaj, że obcięty wielokąt może mieć więcej wierzchołków niż wielokąt nieobcięty.

14.5. Spróbuj opracować program pozwalający korzystać z wszystkich metod podanych dla przestrzeni trójwymiarowej (włącznie z algorytmem malarskim) umożliwiający narysowanie sceny złożonej z kilku obiektów. Będzie potrzebny szybki program sortowania do algorytmu malarskiego. Dla niedużej liczby ścian (powiedzmy do 20) wystarczy zwykły program sortowania bąbelkowego; jednak dla większej liczby będą konieczne takie programy jak Quicksort czy Shellsort.

14.6. Napisz podprogram wykorzystujący prawo Lamberta z korekcją na światło otoczenia, umożliwiający określanie kolorów do cieniowania wielokątów. Pamiętaj o stosowaniu formuły dla każdej z oryginalnych wartości R, G, B i określ wartość maksymalną, której niezależnie od tego, co daje formuła, nie wolno przekroczyć. (Dlaczego?)

15. Spojrzenie na całość

Teraz mamy już dostatecznie dużo informacji o programowaniu do potrzeb grafiki i możemy stworzyć system, który umożliwi stosunkowo łatwe opracowywanie wykresów i obiektów na płaszczyźnie i w trzech wymiarach. System ten nie będzie miał wszystkich możliwości, które ma profesjonalny system graficzny, jednak za jego pomocą można tworzyć i pamiętać rysunki złożone z odcinków kolorowych lub jednobarwnych, a także można przygotować edycję tych rysunków (bezpośrednio w czasie tworzenia lub później). Rysunki mogą być skalowane tak, by uzyskać zgodność z urządzeniem wyjściowym dla nas dostępnym. Można rysować krzywe, elipsy, koła i łuki, a obszary można rozmaicie kreskować. Jeżeli system komputerowy jest wyposażony w wejściowe urządzenie graficzne, takie jak mysz albo digitizer, to można tworzyć rysunki interakcyjnie.

W celu przygotowania oprogramowania systemu, należy podzielić podprogramy na sześć grup.

1. Podprogramy inicjacji systemu graficznego, np. Ustaw-okno, Ustaw-wsPolePrezentacji, WyznaczSkale.
2. Podprogramy przyjmowania danych i realizacji interakcji, np. RysowanieOdcinka i WybórZMenu.
3. Podprogramy modelowania na płaszczyźnie i w trzech wymiarach, np. Okrąg, WypełnianieLinii, CubicFit i Wycinanie.
4. Podprogramy wykonywania transformacji (przesunięcie, obroty, skalowanie) na płaszczyźnie i w trzech wymiarach.
5. Podprogramy obsługi struktur danych, np. Utwórz-Segment, ZapiszDane, OdczytajDane, Dyskop.
6. Podprogramy rysowania, np. RysowanieOdcinka, Łamana, Kreskowanie i Doros.

Do uzyskania kompletnego systemu rysowania obiektów płaskich i obiektów trójwymiarowych są potrzebne podprogramy dla każdej z wymienionych grup. W skład minimalnego systemu do tworzenia rysunków obiektów płaskich, powinny wchodzić następujące grupy programów:

Grupa 1:

Ustaw-Okno	Tablica	10.4
Ustaw-wsPolePrezentacji	Tablica	10.5
WyznaczSkale	Tablica	10.6
ZamianaWspółrz	Tablica	10.7
ObcinanieWielokąta	Tablica	10.9
Przecięcie	Tablica	10.10
Widoczność	Tablica	10.11

Grupa 3:

OkrągJednostkowy	Tablica	8.3
Łuk	Tablica	8.4
StyczneDo2Okręgów	Tablica	8.10

Grupa 4:

Transformacje dwuwymiarowe

Grupa 6:

Łamana	Tablica	11.3
Kreskowanie	Tablica	11.5.

Praktycznie, wszystkie rysunki z tej książki mogłyby być narysowane za pomocą systemu graficznego, zawierającego tylko tę małą grupę podprogramów. Oczywiście więcej wysiłku wymaga eksploatacja systemu minimalnego, niż systemu o pełniejszym zbiorze funkcji. Na przykład w systemie bez możliwości interakcyjnych, wszystkie dane dla rysunków powinny być przygotowane poza systemem i wprowadzone ręcznie do systemu. Programowanie w grafice komputerowej najlepiej zacząć od tego najprostszego systemu i stopniowo dodawać nowe podprogramy w miarę pojawiających się potrzeb. W ten sposób można dysponować pracującym systemem już po kilku godzinach programowania. W miarę upływu czasu będą potrzebne nie tylko wymienione programy, ale i wiele własnych, zwłaszcza do modelowania takich kształtów jak kwadraty, prostokąty, sześciany i inne regularne obiekty dwuwymiarowe i trójwymiarowe. Aby system nie rozrastał się, może okazać się celowe rozdzielenie systemów rysowania obiektów dwuwymiarowych i trójwymiarowych, a nawet wydzielenie specjalnych programów, które będą tworzyły dane o obiektach niezależnie od rysunków.

Do nowoprojektowanych systemów należy dołączyć testy dla niepoprawnych danych i złych wywołań podprogramów, w liczbie większej niż pokazano w szkicowych podprogramach i algorytmach. Przykładowym zabezpieczeniem jest zatrzymanie programu wówczas, gdy jeden z podprogramów wykryje błąd. Takie rozwiązanie jest przyjęte w podprogramach Ustaw-okno i Ustaw-wsPolePrezentacji. Sposób ten jest dobry, ale nie jest zbyt wygodny; lepszym rozwiązaniem jest dynamiczne korygowanie błędów przez użytkownika, bez konieczności przejścia programu do awaryjnego zakończenia. Ma to znaczenie zwłaszcza w przypadku, interakcyjnego wprowadzania rysunków i innych danych.

Dalsza lektura

W miarę rozwoju swojego systemu, Czytelnik będzie mógł wykorzystywać go nie tylko do tworzenia rysunków, ale i do testowania nowych pomysłów i metod grafiki komputerowej. Grafika komputerowa wciąż się rozwija, a jej zagadnienia są poruszane w wielu specjalistycznych pismach. Po zapoznaniu się z tą książką, Czytelnik nie powinien mieć większych trudności w zrozumieniu artykułów technicznych zamieszczanych w tych pismach.

W szczególności są dostępne trzy pisma, w których są zamieszczane najnowsze osiągnięcia z dziedziny grafiki komputerowej. Są to: *Computer Graphics Forum* publikowane przez European Association for Computer Graphics (Eurographics), oraz *Computer Graphics* i *Transactions on Computer Graphics* publikowane przez Special Interest Group on Computer Graphics w ramach Association for Computing Machinery (ACM SIGGRAPH). Zamieszczane w tych pismach artykuły stanowią podstawowe źródło informacji dla każdego zainteresowanego rozwijaniem swoich umiejętności w tej dziedzinie. Bez posiadania specjalnych kwalifikacji, można zostać członkiem obu tych stowarzyszeń, które corocznie organizują konferencje międzynarodowe - Eurographics, w różnych krajach Europy i SIGGRAPH w USA. Kopie materiałów z tych konferencji są bardzo wartościowym źródłem informacji. Wiele bibliotek publicznych i uniwersyteckich posiada te czasopisma i materiały konferencyjne.

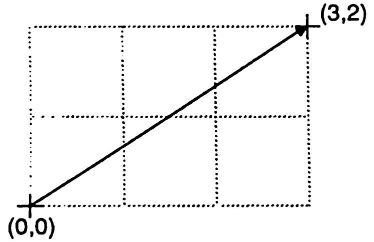
Przy dalszym studiowaniu grafiki komputerowej pomocne będą trzy książki: *Principles of Interactive Computer Graphics* (drugie wydanie) - W. M. Newman i R. F. Sproull (McGraw-Hill Book Company, 1979); *Fundamentals of Interactive Computer Graphics* - J. D. Foley i A. van Dam (Addison - Wesley, 1982); *Procedural Elements for Computer Graphics* - D. F. Rogers (McGraw-Hill Book Company, 1985). Książki te można traktować jako wprowadzające, mimo że w każdej z nich jest stosowane inne podejście do zagadnień grafiki. Książka Rogersa zawiera wiele użytecznych algorytmów i przykładów, które mogą być używane do testowania podprogramów, napisanych przez Czytelnika po przerobieniu tekstu. Przypadkowo Rogers używa również tego samego pseudokodu, który przyjęto w tej książce (w istocie jest to wersja Basicu stosowana w komputerach biurowych Tektronix 4054A).

Grafiki komputerowej można się nauczyć jedynie przez praktykę. Książki i czasopisma mogą pomóc, ale pewność, że się rozumie podstawy, można uzyskać jedynie pisząc programy i testując je na różnych przykładach. Już Konfucjusz mówił, że:

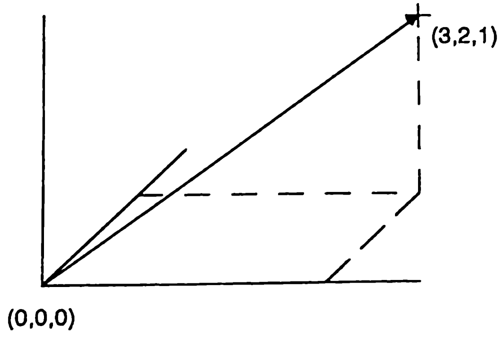
Słyszę i zapominam,
Widzę i pamiętam,
Wykonuję i rozumiem.

Algebra wektorów i macierzy

Wektory są to uporządkowane zbiory liczb, które w dziedzinie grafiki mogą być używane do reprezentowania odcinków mających zarówno długość jak i kierunek. Chociaż liczby (*elementy składowe*) definiujące wektory mogą reprezentować wiele różnych danych, to w grafice określają współrzędne położenia w kartezjańskim systemie współrzędnych zdefiniowanym tak, jak podano w rozdz. 6. Wektor (3,2) może reprezentować odcinek w przestrzeni dwuwymiarowej zaczepiony w początku układu współrzędnych (0,0) i o końcu w punkcie (3,2) (rys. A1.1). Podobnie wektor (3,2,1) jest odcinkiem w przestrzeni trójwymiarowej, narysowanym od początku układu współrzędnych (0,0,0) do punktu (3,2,1) (rys. A1.2).



Rys. A1.1. Wektor łączący punkty (0,0) i (3,2)

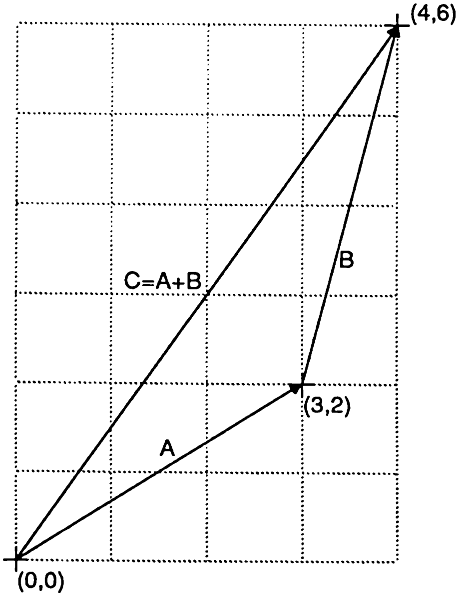


Rys. A1.2. Wektor łączący punkty (0,0,0) i (3,2,1)

Przekształcenia algebraiczne i operacje arytmetyczne mogą być wykonywane na wektorach prawie tak, jak gdyby były one zwykłymi liczbami. Na przykład wektory można: dodawać lub odejmować, mnożyć przez liczbę lub mnożyć przez inny wektor. Nie jest możliwe dzielenie wektorów.

Dodawanie i odejmowanie wektorów

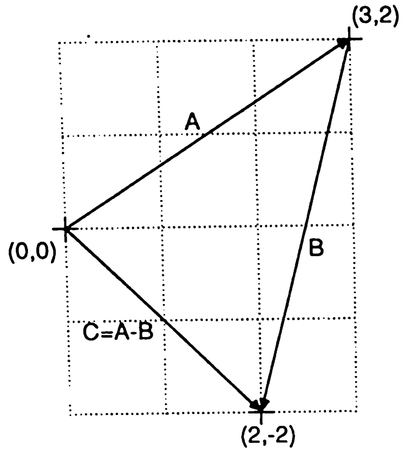
Wektor (X1,Y1) może być dodany do wektora (X2,Y2); w wyniku otrzymuje się wektor (X1 + X2, Y1 + Y2). Geometryczna interpretacja wektorów jest pokazana na rys. A1.3. Wektor A reprezentowany przez (3,2) jest dodawany do wektora B reprezentowanego przez (1,4); w wyniku otrzymuje się wektor C, (4,6).



Rys. A1.3. Dodawanie wektorów

Operacja dodawania A + B jest równoważna rysowaniu odcinka od początku układu współrzędnych do punktu przesuniętego o 3 jednostki w kierunku osi X i dwie jednostki w kierunku osi Y, a następnie odcinka zaczynającego się w tym punkcie i kończącego się w punkcie odległym o jednostkę w kierunku X i cztery jednostki w kierunku Y. Operacja A + B daje ten sam wynik co operacja B + A. Odjęcie wektora B od wektora A (A – B) daje w wyniku wektor (X1 – X2, Y1 – Y2).

Na rys. A1.4 podano geometryczną ilustrację tego działania. Wektor B (1,4) jest odejmowany od wektora A (3,2), co daje w wyniku wektor A – B, (2,–2). Operacja A – B jest równoważna rysowaniu odcinka od końca odcinka B. do końca odcinka A. Jak



Rys. A1.4. Odejmowanie wektorów

w wzwykłej arytmetyce, poza przypadkiem A = B, wektor A – B jest różny od wektora B – A. W istocie, A – B ma tę samą długość co B – A, tylko zwroty odcinków są przeciwne.

Mnożenie przez skalar

Jeżeli wektor jest mnożony przez dodatnią liczbę, to zmienia się długość wektora; a więc S * (X,Y,Z) daje nowy wektor (S*X, S*Y, S*Z). Jeżeli S jest zawarte w przedziale (0,1), to wynikowy wektor jest krótszy (nie zmienia się, gdy S = 1). Jeżeli S jest większe niż 1, to długość wektora zwiększa się. Taki sam rezultat otrzymuje się przy mnożeniu wektora przez S ujemne; zmienia się tylko zwrot wektora.

Długość wektora

Długość wektora obliczamy korzystając z twierdzenia Pitagorasa. Długość wektora (X1,Y1) jest określona wyrażeniem:

$$\text{Długość} = \sqrt{X1^2 + Y1^2}.$$

Długość wektora (X1,Y1,Z1) jest określona następująco:

$$\text{Długość} = \sqrt{X1^2 + Y1^2 + Z1^2}$$

Mnożenie wektorów

Są dwa sposoby mnożenia wektorów przez siebie. Jeden z nich to tzw. *iloczyn skalarny*, o zapisie w postaci: A · B. Wynikiem mnożenia skalarnego wektorów na płaszczyźnie

lub w przestrzeni jest liczba. Drugi sposób mnożenia to tzw. *iloczyn wektorowy*, o zapisie w postaci: $A \times B$. Wynikiem takiego mnożenia jest wektor; iloczyn wektorowy nie jest definiowany w przestrzeni dwuwymiarowej.

Iloczyn skalarny

Iloczyn skalarny dwóch niezerowych wektorów w przestrzeni dwu- lub trójwymiarowej jest liczbą rzeczywistą określoną następująco:

$$A \cdot B = \text{Długość}(A) * \text{Długość}(B) * \cos(Q);$$

gdzie Q jest kątem między wektorami.

Dla wektorów A (XA,YA,ZA) i B (XB,YB,ZB), z których żaden nie ma długości zerowej, iloczyn skalarny oblicza się na podstawie składowych wektorów w następujący sposób:

$$A \cdot B = XA * XB + YA * YB + ZA * ZB.$$

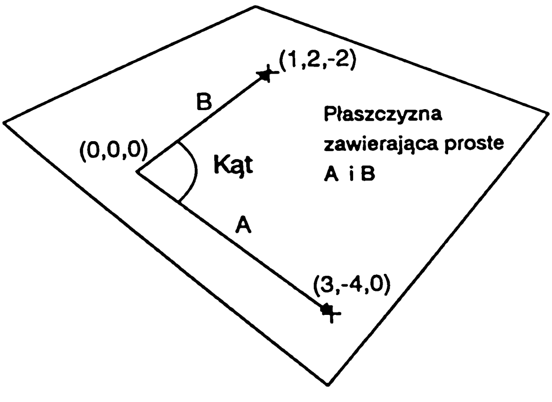
Kąt Q między dwoma wektorami A i B wyznacza się następująco:

$$\cos(Q) = (A \cdot B)/(\text{Długość}(A) * \text{Długość}(B)) =$$
$$= (XA * XB + YA * YB + ZA * ZB)/L;$$

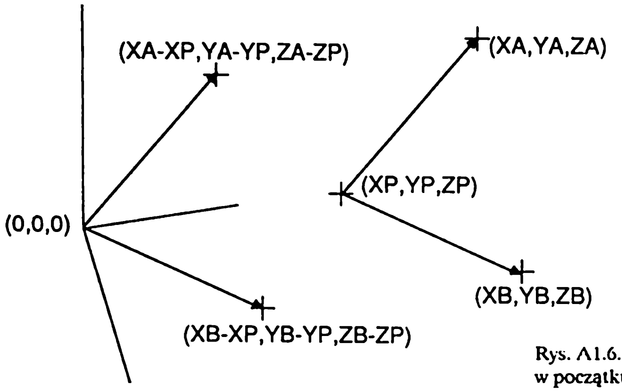
gdzie $L = \sqrt{((XA^2 + YA^2 + ZA^2) \cdot (XB^2 + YB^2 + ZB^2))}$.

Na przykład, jeżeli A = (3,–4,0) i B = (1,2,–2) to:
A · B = 3 – 8 + 0 = –5.
Długość (A) = 5.
Długość (B) = 3.
cos (Q) = –5/(5 * 3) = –1/3.
$Q = 109.471^{\circ}$.

W tym przypadku kąt jest mierzony w płaszczyźnie określonej przez dwa wektory tak, jak pokazano na rys. A1.5. W celu obliczenia kąta między dwoma odcinkami, które zaczynają się we wspólnym punkcie P nie będącym początkiem układu współrzędnych (rys. A1.6), najpierw należy przesunąć punkty odcinków: początkowy do początku



Rys. A1.5. Kąt między dwoma wektorami jest mierzony w płaszczyźnie, w której leżą te wektory



Rys. A1.6. Iloczyn skalarny należy obliczać w początku układu współrzędnych

układu współrzędnych i odpowiednio końcowe, następnie obliczyć iloczyn skalarny. Jeżeli więc P = (XP,YP,ZP) jest punktem początkowym odcinków 1 i 2, a punktami końcowymi są (XA,YA,ZA) i (XB,YB,ZB), to do obliczenia iloczynu skalarnego należy użyć nowych wektorów

$$(XA - XP, YA - YP, ZA - ZP) \text{ i } (XB - XP, YB - YP, ZB - ZP).$$

Jeżeli dla dwóch, niezerowych wektorów A i B iloczyn skalarny $A \cdot B = 0$, to wektory A i B są prostopadłe do siebie, ponieważ $\cos(90°) = 0$. Prostopadłość jest przedstawiona na rys. 6.2.

Iloczyn wektorowy

Iloczyn wektorowy $A \times B$ definiuje wektor C, który jest prostopadły do wektorów A i B; co oznacza, że $A \cdot C = 0$ i $B \cdot C = 0$.
Jeżeli A jest wektorem (XA,YA,ZA), a B jest wektorem (XB,YB,ZB), to $A \times B$ wyznacza się za pomocą wyrażenia:

$C = A \times B = (XC,YC,ZC);$

gdzie:
XC = YA * ZB – ZA * YB
YC = ZA * XB – XA * ZB
ZC = XA * YB – YA * XB.
Na przykład, jeżeli A = (5,–3,1) i B = (–1,–1,2), to $C = A \times B$ = (XC,YC,ZC)
i
XC = (–3 * 2) – (1 * (–1)) = –6 + 1 = –5
YC = (1 * (–1)) – (5 * 2) = –1 – 10 = –11
ZC = (5 * (–1)) – ((–3) * (–1)) = –5 – 3 = –8.
Stąd: C = (–5,–11,–8). Wektor ten jest prostopadły do A i B, co możemy sprawdzić przez obliczenie iloczynów skalarnych $A \cdot C$ i $B \cdot C$.

$$A \cdot C = XA * XC + YA * YC + ZA * ZC =$$
$$= (5 * (-5) + ((-3) * (-11)) + (1 * (-8)) = -25 + 33 - 8 = 0$$

$$B \cdot C = XB * XC + YB * YC + ZB * ZC =$$
$$= (-1 * (-5)) + ((-1) * (-11)) + (2 * (-8)) = 5 + 11 - 16 = 0$$

Płaszczyzna jest zdefiniowana przez dowolne trzy oddzielne punkty (nie leżące na jednej prostej), inaczej przez dowolne dwa odcinki łączące te punkty. Własność prostopadłości cechująca iloczyn wektorowy jest szczególnie ważna w grafice komputerowej, w której zachodzi konieczność znalezienia wektorów, prostopadłych (normalne) do powierzchni. Częstym wymaganiem jest jednostkowa długość normalnych; otrzymujemy ją dzieląc każdą składową wektora przez długość wektora. Normalna (–3,–6,–2) ma długość L:

$$L = \sqrt{(9 + 36 + 4)} = 7.$$

To oznacza, że normalny wektor jednostkowy ma postać (–3/7,–6/7,–2/7). Można to potwierdzić obliczając jego długość:

$$Lr = \sqrt{(9/49 + 36/49 + 4/49)} = 1.$$

W przeciwieństwie do iloczynu skalarnego, iloczyn wektorowy zależy od kolejności, w jakiej wektory są przez siebie mnożone. Innymi słowy, AxB to nie to samo co BxA. Dokładnie biorąc $A \times B = -(B \times A)$ i zwroty obu wektorów są przeciwne. Warto to sprawdzić na przykładzie wektorów A = (5,–3,1) i B = (–1,–1,2), które już były stosowane w innych obliczeniach.

Dodatek 2

Użytkowanie podprogramów

Z powodów podanych we wstępie, wszystkie podprogramy zamieszczone w książce zostały napisane w pseudokodzie, który nie może być wykonywany bezpośrednio w komputerze. W praktyce każdy programista pracuje, korzystając z języka programowania takiego, jak Basic, Pascal lub C i będzie musiał przekształcić programy w pseudokodzie na postać właściwą dla danego języka. Podane podprogramy mają następującą strukturę:

Subroutine Nazwa(Parametry wejściowe; Parametry wyjściowe)
Właściwy pseudokod podprogramu
End podprogramu

Parametry wejściowe są podawane przez program do podprogramu; parametry wyjściowe są wynikiem pracy podprogramu, przekazywanym do programu głównego. Te dwie grupy parametrów są oddzielone średnikiem. Czytelnicy, którzy programują tylko w standardowym Basicu nie znają zapewne metody przekazywania danych przez parametry. Powinni jednak zauważyć, że parametry są to zmienne puste (ang. dummy), mające nazwy używane tylko w samym podprogramie. Tak więc podprogram ObliczKąt (tabl. 8.8) ma następujące wejściowe, zmienne puste: XC, YC, X, Y, Znacznik oraz jedną wyjściową zmienną pustą A. Do użycia tego podprogramu należy podać nazwy zmiennych rzeczywistych (lub zamiast zmiennych wejściowych - nazwy stałych), używanych w pozostałej części programu. Jeśli wywołamy ten podprogram następująco:

Call ObliczKąt(Xp,Yp,Xkon,Ykon,0;SpecjalnyKąt),

to podprogram użyje Xp jako XC, Yp jako YC, Xkon jako X, Ykon jako Y, 0 (stała zero) jako Znacznika. Podstawi także obliczoną przez siebie wartość A do zmiennej SpecjalnyKąt. Program może teraz dowolnie użyć tego wyniku, określającego kąt między osią X, a prostą łączącą punkty (Xp, Yp) i (Xkon, Ykon).

Zaletą takiego użycia podprogramu jest to, że można dowolnie wiele razy wywołać podprogram dla różnych nazw i wartości parametrów wejściowych, bez obawy - jak to często występuje w Basicu - że użyjemy nazwy zmiennej, która już występowała w innym kontekście.

Wewnątrz podprogramów wyróżniamy *zmienne lokalne* i *globalne*. Zmienną nazywamy lokalną jeśli istnieje tylko wewnątrz podprogramu; jeśli zmienna ma wpływ na zmienne o tej samej nazwie poza podprogramem, to jest zmienną globalną. Rozróżniamy także *liczby całkowite* (**integer**) i *liczby rzeczywiste* (**real**). W Basicu nie wyróżniamy tych liczb, chociaż wiele współczesnych interpretatorów tego języka działa szybciej, jeśli zostanie to zrobione. W językach strukturalnych wysokiego poziomu, np. w Pascalu i C zmienne muszą mieć (wstępnie przed ich użyciem) określony typ.

Wiele wersji Basicu nie pozwala na użycie zmiennych pustych; aby więc przetłumaczyć podane podprogramy należy użyć różnych nazw zmiennych w podprogramie (tabl. A2.1), lub zastosować sztuczkę pokazaną w tabl. A2.2. Pomysł polega na tworzeniu specjalnego zbioru zmiennych tablicowych, np. T() i T$(), w których zapamietujemy wartości dowolnej zmiennej, mającej tę samą nazwę co zmienna w podprogramie. Po opuszczeniu podprogramu zmienne wynikowe otrzymują nowe nazwy, a innym zmiennym zostają przywrócone ich własne nazwy. Jeśli Czytelnik zakoduje przykład z tabl. A2.2 i przypisze różne wartości zmiennym, to łatwo stwierdzi jak ten program działa. Tę sztuczkę można stosować również w innych programach.

Konstrukcja, nie spotykana w Basicu, to instrukcja **Case**. Sposób, w jaki była zastosowana w książce, odbiega nieco od sposobu jej użycia w języku C. Nasza konstrukcja **Case** ma postać:

```
Case of num:
    1: X = 5
    2: X = 26.5
    3: X = –14
End of case.
```

W tym przypadku działanie programu zależy od wartości zmiennej *num*. Jeśli *num* = 1, to X przyjmuje wartość 5 i program przechodzi do instrukcji następującej za **End**; jeśli *num* = 2, to X przyjmuje wartość 26.5 i program przechodzi do instrukcji następującej za **End**; jeśli *num* = 3, to X otrzymuje wartość –14 i program kontynuuje swoje działanie. Taką konstrukcję łatwo jest oprogramować w Basicu korzystając z instrukcji **GOTO obliczanego** (jeśli jest dostępna) lub **GOTO zwykłego**.

Przykład

```
150 IF NUM = 1 THEN X = 5: GOTO 200
160 IF NUM = 2 THEN X = 26.5: GOTO 200
170 X = –14
200 REM - DALSZA CZĘŚĆ PROGRAMU
```

W tablicy A2.3 pokazano jak można użyć **Case** w Pascalu.

W tablicy A2.4 pokazano przykład programu w Basicu Microsoft dla komputera Apple MacIntosh. Program służy do testowania podprogramu obracania przekroju do uzyskania figury trójwymiarowej. Program stosuje pomysły z rozdz. 13 i pokazuje jak można użyć niektórych z podprogramów opisanych w książce do rozwiązywania zadań praktycznych.

Tablica A2.1. Program obcinania w Basicu

```
   1  REM - testowanie procedury obcinania w Basicu
  10  W1 = 0: W2 = 1200: W3 = -600: W4 = 600
  20  N = 3
  30  DIM A1(N,2),O1(2*N,2),O2(2*N,2)
  40  REM - odczytaj współrzędne wielokąta i wstaw je do O1()
  50  FOR J = 1 TO N
  60  READ A1(J,1),A1(J,2)
  70  O1(J,1) = A1(J,1): O1(J,2) = A1(J,2)
  80  NEXT J
  90  DATA 600,1000,1400,800,1400,-600
  95  REM - sprawdź po kolei wszystkie krawędzie
 100  FOR E =1 TO 4: GOSUB 1000; NEXT E
 190  REM - wydrukuj wyniki
 200  FOR J = 1 TO C-1; PRINT O2(J,1), O2(J,2): NEXT J
 999  END
1000  REM - podprogram obcinania
1010  C =1
1020  S1 = O1(N,1): S2 = O1(N,2)
1040  FOR J = 1 TO N
1050  P1 = O1(J,1): P2 = O1(J,2)
1070  X1 = P1: Y1 = P2: GOSUB 3000
1080  IF V = 1 THEN 1200
1090  X1 = S1: Y1 = S2: GOSUB 3000
1100  IF V = 1 THEN 1110
1105  GOTO 1400
1110  X1 = S1: Y1 = S2: X2 = P1: Y2 = P2: GOSUB 1500
1115  IF R>1 OR R< =0 THEN 1400
1120  O2(C,1) = X: O2(C,2) = Y
1130  C = C + 1
1135  GOTO 1400
1200  X1 = S1: Y1 = S2: GOSUB 3000
1210  IF V = 1 THEN 1300
1220  X1 = S1: Y1 = S2: X2 = P1: Y2 = P2: GOSU 1500
1225  IF R>1 OR R< =0 THEN 1250
1230  O2(C,1) = X: O2(C,2) = Y
1240  C = C + 1
1250  O2(C,1) = P1: O2(C,2) = P2: C = C + 1
1270  GOTO 1400
1300  O2(C,1) = P1: O2(C,2) = P2: C = C + 1
1400  S1 = P1: S2 = P2
1410  NEXT J
```

```
1420  FOR J = 1 TO C - 1
1430  O1(J,1) = O2(J,1): O1(J,2) = O2(J,2)
1450  NEXT J
1455  N = C - 1
1460  RETURN
1500  REM - procedura przecięć
1510  ON E GOTO 2000,2200,2400,2600
2000  R = (W4 - Y1) / (Y2 - Y1)
2010  X = X1 + (X2 - X1) * R
2020  Y = W4
2030  RETURN
2200  R = (W2 - X1) / (X2 - X1)
2210  Y = Y1 + (Y2 - Y1) * R
2220  X = W2
2230  RETURN
2400  R = (W3 - Y1) / (Y2 - Y1)
2410  X = X1 + (X2 - X1) * R
2420  Y = W3
2430  RETURN
2600  R = (W1 - X1) / (X2 - X1)
2610  Y = Y1 + (Y2 - Y1) * R
2620  X = W1
2630  RETURN
3000  REM - podprogram sprawdzania widoczności
3010  ON E GOTO 3020,3040,3060,3080
3020  X0 = W1: Y0 = W4: X9 = W2: Y9 = W4
3030  GOTO 3090
3040  X0 = W2: Y0 = W4: X9 = W2: Y9 = W3
3050  GOTO 3090
3060  X0 = W2: Y0 = W3: X9 = W1: Y9 = W3
3070  GOTO 3090
3080  X0 = W1: Y0 = W3: X9 = W1: Y9 = W4
3090  V1 = (X9 - X0) * (Y9 - Y1) - (Y9 - Y0) * (X9 - X1)
3100  V = 1
3110  IF V1 < = 0 THEN V = 0
3120  RETURN
```

Tablica A2.2. Wywołanie podprogramów i przykładowe wyniki

```
100  REM - program ilustrujący jak wywoływać podprogramy
105  REM - podprogram ma postać NIC(J;A), przyjmuje J, dajc A
110  DIM T(5), T$(5)
120  B$ = "string1"
130  A = 10.4
```

```
 140  FOR J = 1 TO 2
 150  A = A * J
 160  REM - przygotuj wywołania podprogramu przez zapamiętanie
 165  REM - danych z programu
 170  T(1) = A: T(2) = J: T$(1) = B$
 180  PRINT "W programie: A = ";A;"J = ";J;"B$ = ";B$
 190  REM - wywołaj podprogram NIC z użyciem tych zmiennych
 200  GOSUB 1000
 210  PRINT "Program przywrócił wartość A = ";A
 220  REM - powrót do poprzednich wartości
 230  A = T(1): J = T(2): B$ = T$(1)
 240  NEXT J
 250  END
1000  REM - podprogram NIC(J;A)
1010  B$ = "string2"
1020  J = J * 2
1030  A = J + 2
1040  PRINT " W podprogramie NIC: A = ";A;"J = ";J;"B$ = ";B$
1050  RETURN
```

Wydruk z programu

```
W programie: A = 10.4 J = 1 B$ = string1
W podprogramie NIC: A = 4 J = 2 B$ = string2
Program zwrócił wartość A = 4
W programie: A = 20.8 J = 2 B$ = string1
W podprogramie NIC: A = 6 J = 4 B$ = string2
Program zwrócił wartość A = 6
```

Tablica A2.3

```
Program rotmattest; {część testu w języku Pascal, podprogramu rotmat}
type
  mat4by4 = array[1..4, 1..4] of real
var
  i: integer;
  a: real;
  mat: mat4by4;
procedure rotmat (axstr:char; a:real; var mtrx: mat4by4);
{ procedura liczenia macierzy obrotu 4×4 }
var
  c, s: real;
  i, j: integer;
begin
  a := a *3.1415926 / 180; { przekształcenie kąta na radiany }
  c := cos(a);
```

```
s := sin(a);
for i := 1 to 4 do
  for j := 1 to 4 do
    begin
      if i = j then
        mtrx[i,j] := 1
      else
        mtrx[i,j] := 0
      end;
    case axstr of
      'x':begin
            mtrx[2,2] := c;
            mtrx[2,3] := -s;
            mtrx[3,2] := s;
            mtrx[3,3] := c;
          end; { obrotu wokół osi X }
      'y':begin
            mtrx[1,1] := c;
            mtrx[1,3] := s;
            mtrx[3,1] := -s;
            mtrx[3,3] := c;
          end; { obrotu wokół osi Y }
      'z':begin
            mtrx[1,1] := c;
            mtrx[1,2] := s;
            mtrx[2,1] := -s;
            mtrx[2,2] := c;
        end; { obrotu wokół osi Z }
      end; { instrukcji case }
end; { podprogramu rotmat }
```

Tablica A2.4. Program w Basicu MacIntosh

```
'Program w MS Basicu dla komputera MacIntosh do tworzenia obiektów
'przez obrót krzywej i do pokazania ich w perspektywie. Profil do
'obracania musi znajdować się w górnej półpłaszczyźnie x-y; jego
'pierwszy punkt musi znajdować się na górze (zauważmy, że w  komputerze MacIntosh
'początek układu współrzędnych znajduje się w lewym górnym rogu
'ekranu)
'ustal wielkośc ekranu i znajdź środek
      w1 = 0: w2 = 0: w3 = 511: w4 = 341
      w5% = INT(w3/2):w6% = INT(w4/2)        'środek ekranu
      WINDOW1,,(w1,w2)-(w3,w4),3              'specyficzne dla MacIntosh
'deklaracje i podanie wartości pewnych stałych
```

```
    DIM Px(18),Py(18),FPx(256),FPy(256),FPz(256)
    DIM Poly(40),kwadrściany(480,5),trojkściany(48,4),XP(256),YP(256)
    pi = 3.1415926*
    f = PI/180                              'współczynnik przeliczenia na radiany
    dist = 250                              'odległość od obserwatora
'odczytaj dane profilu i liczbę ścian
    READ lscian%                            'całkowita liczba ścian
    READ lpunktow%                          'całkowita liczba punktów
    IF lpunktow% > 20 THEN PRINT "Dopuszcza się tylko 20 punktów":
STOP
    READ PSx,PSy                            'odczytaj górny punkt
    FOR J% = 1 TO lpunktow%-2               'punkty pośrednie
      READ Px(J%),Py(J%)
    NEXT J%
    READ PEx,PEy                            'punkt dolny
'określ współrzędne wszystkich punktów
    FPx(1) = PSx: FPy(1) = PSy: FPz(1) = 0 'punkt górny
    FOR J% = 2 TO lpunktow% - 1             'pierwszy przekrój
      FPx(J%) = Px(J%-1): FPy(J%) = Py(J%-1): FPz(J%) = 0
    NEXT J%
    aktnum% = lpunktow%
    kat = 360/lscian%*f                     'przelicz kąt na radiany
    FOR K% = 1 TO lscian%                   'pozostałe przekroje
      FOR J% = 2 TO lpunktow% - 1
        FPx(aktnum%) = INT(Px(J% - 1) * COS(K% * kat))
        FPy(aktnum%) = Py(J% - 1)
        FPz(aktnum%) = INT(Px(J% - 1) * SIN(K% * kat))
        aktnum% = aktnum% + 1
      NEXT J%
    NEXT K%
      FPx(aktnum%) = PEx:FPy(aktnum%) = PEy:FPz(aktnum%) = 0
      ostatni% = aktnum%                    'wszystkie punkty zakończono
     'ustal listę scian
     'górne ściany trójkątne
    FOR numsciany% = 1 TO lscian%
      pt1% = 2 + (numsciany% - 1) * (lpunktow% - 2)
      pt2% = 2 + numsciany% * (lpunktow% - 2)
      trojksciany(numsciany%,1) = pt1%
      trojksciany(numsciany%,2) = 1
      trojksciany(numsciany%,3) = pt2%
      trojksciany(numsciany%,4) = pt1%
    NEXT numsciany%
        'dolne ściany trójkątne
```

```
FOR numsciany% = 1 TO lscian%
   pt1% = (lpunktow% - 1) + (numscian% - 1) * (lpunktow% - 2)
   pt2% = (lpunktow% - 1) + numscian% * (lpunktow% - 2)
   aktnum% = numscian% + lscian%
   trojksciany(aktnum%,1) = pt1%
   trojksciany(aktnum%,2) = pt2%
   trojksciany(aktnum%,3) = ostatni%
   trojksciany(aktnum%,4) = pt1%
NEXT numsciany%
ltrojkscian% = aktnum%
'wszystkie ściany kwadratowe
nkwadrscian% = 0
FOR strona% = 0 TO lscian% - 1
  FOR warstwa% = 1 TO lpunktow% - 3
    nkwadrscian% = nkwadrscian% + 1
    start% = (warstwa% + 1) (lpunktow% - 2) * strona%
    kwadrsciany(nkwadrscian%,1) = start%
    kwadrsciany(nkwadrscian% 2) = start% + (lpunktow% - 2)
    kwadrsciany(nkwadrscian%,3) = start% + (lpunktow% - 2) + 1
    kwadrsciany(nkwadrscian%,4) = start% + 1
    kwadrsciany(nkwadrscian%,5) = start%
   NEXT warstwa%
  NEXT strona%              'lista ścian kompletna
  lkwadrscian% = nkwadrscian%
  'umieśc wszystkie punkty we właściwej perspektywie; obserwator
  'w punkcie (0, 0, -dist); przenieś na ekran
  FOR J% = 1 TO ostatni%
          = dist/(FPz(J%) + dist)
   XP(J%) = INT(FPx(J%)*perspekt + 5) + w5%
   YP(J%) = INT(PFy(J%)*perspekt + 5) + w6%-100
  NEXT J%
  'narysuj wszystkie trójkątne ściany
  FOR J% = 1TO ltrojkscian%
   licz% = trojksciany(J%,1)
   CALL MOVETO(XP(licz%),YP(licz%))
   FOR K% = 2 TO 4
    licz% = trojksciany(J%,K%)
    CALL LINETO(XP(licz%),YP(licz%))
   NEXT K%
  NEXT J%
  'narysuj wszystkie kwadratowe sciany
  FOR J% = 1TO lkwadrscian%
   licz% = kwadrsciany(J%,1)
```

```
 CALL MOVETO(XP(licz%),YP(licz%))
 FOR K% = 2 TO 5
  licz% = kwadrsciany(J%,K%)
  CALL LINETO(XP(licz%),YP(licz%))
 NEXT K%
NEXT J%
hold:IF INKEY$ ="" THEN hold      'utrzymuj obraz do naciśnięcia Enter
END
DATA 8,7
DATA 0,200,20,200,20,140,40,120,40,60,20,40,0,40
```

Skorowidz

WYDAWNICTWA NAUKOWO-TECHNICZNE
ul. Mazowiecka 2/4, 00-048 Warszawa
tel. 26-72-71 do 79
Dział Upowszechniania i Sprzedaży
tel. 27-56-87

WNT Warszawa 1990.
Wyd. I. Ark. wyd. 13,5. Ark. druk. 12,0
Format B5. Et/82518/WNT
Bielskie Zakłady Graficzne
zam. 1594/90

ISBN 83-204-1229-3